PROYECTO FIN DE CARRERA

UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA INDUSTRIAL TÉCNICA: MECÁNICA

Implementación de una aplicación para la visualización de resultados en un programa de cálculo matricial

Departamento de mecánica de medios continuos y teoría de estructuras

Autor: Alberto Madueño Asensio
Tutor: D.Dr.Carlos Santiuste Romero

Agradecimientos

A mi tutor, Carlos, cuya paciencia, dedicación y motivación, siempre desde la mejor actitud y comprensión en sus enseñanzas, han hecho posible la realización de este proyecto.

A creador del programa original, que se mostró abierto y receptivo a ayudarme en la medida de lo posible.

A mis padres y hermanos por todo el apoyo aportado.

A mis amigos, en especial a Pedro y Alfonso, que se ofrecieron a ayudarme en todo lo que estuviera en su mano.

INDICE

INDICE DE FIGURAS

1. INTRODUCCIÓN

1.1. Motivación

La sociedad a lo largo de la historia, ha ido desallorándose y evolucionando tecnológicamente gracias a diversas técnicas de investigación e innovación que van siendo requeridas poco a poco en todos y cada uno de los aspectos que generan la sociedad.

Los componentes de esta sociedad se benefician de todas estas mejoras en su día a día siendo la mayoría de ellos poco conscientes del trabajo que todo ello conlleva, la complejidad y profundidad que dichos estudios conllevan.

De cara al ciudadano de a pie, pocas son las innovaciones que llaman lo suficientemente la atención como para ser admiradas. Ellas suelen ser las que dan uso cotidiano además de ser llamativas a la vista del ojo humano.

Por todo ello estos desarrollos industriales que más suscitarán la curiosidad de la sociedad serán las construcciones hidráulicas(embalses), infraestructuras destinadas al transporte, siendo un claro ejemplo las carreteras o las vías para ferrocarriles o los medios de transporte que secundarán dichas estructuras como aviones, coches, trenes… etc y por último, siendo uno de los más vistosos, los edificios.

Figura 1.1. Puente

Si hay algo que tienen en común todas las infraestructuras citadas, es la estructura básica de la que parten. En todas ellas se elabora una estructura inicial que se calculará de una manera parecida en todas para su posterior desarrollo en profundidad.

Estos cálculos van dirigidos en función de dos objetivos claros y concisos:

- El primero y más importante de todos será la seguridad. A la hora de llevar a cabo una cierta composición, ya sea destinada al uso de personas o no, requiere de unos cálculos que estén sometidos a unos coeficientes de seguridad que aseguren el buen funcionamiento de la misma y un margen de libertad que nos afiance que no va a ocurrir nada en el caso de que se apliquen unos esfuerzos superiores a los que se puedan predecir.

- El segundo objetivo será el tema económico, en el cual se intentará acercar lo más posible a las exigencias que requiera el mercado en ese momento. Aquí cobra gran importancia la minimización de la materia prima debido a su elevado coste, en proceso de aumento.

Con estos dos objetivos, en ocasiones inversamente proporcionales, se intenta trabajar meticulosamente para intentar llegar a una resolución de compromiso en que se consiga un coeficiente de seguridad/precio de coste lo más alto posible que nos proporcione una estructura lo más competitiva posible.

Estas estructuras, en términos generales estarán sometidas a fuerzas estáticas y dinámicas.

Para el estudio de deformaciones y esfuerzos bajo cargas estáticas existen softwares informáticos como:

- Software simple de cálculo matricial: Programas que permiten calcular estructuras articuladas y reticuladas en 2 y 3 dimensiones atacadas con cargas estáticas de diversas formas que opera con cálculos matriciales y de sencillo manejo, como por ejemplo Ed-Tridim.

- Software complejo basado en normativa: Con la diferencia del Software simple de cálculo matricial que se ha sido concebido para realizar el cálculo y dimensionamiento de estructuras de hormigón armado y metálicas, sometidas a acciones horizontales y verticales, para viviendas, edificios y proyectos de obra civil basado en normativas y de un uso bastante más complejo, como por ejemplo el CYPECAD.

Aunque con los análisis estáticos, se pueden dimensionar correctamente la mayoría de las estructuras, siempre existen excepciones que requieren un análisis dinámico con mayor precisión que se asemeje más a la realidad.

Por otra parte, los avances tecnológicos, han generado un desarrollo de estructuras más precisas y susceptible a flexión que generan una mayor importancia en estas cargas dinámicas.

Estas cargas dinámicas pueden venir generadas por:

- El esfuerzo que puede generar el viento y las variaciones del mismo u otras condiciones meteorológicas.
- Cargas oscilantes por el frecuente paso de maquinaria pesada.
- Cargas instantáneas producidas de forma accidental.

Estos son posibles ejemplos, pero las cargas dinámicas pueden venir por una infinidad de causas.

Para el análisis de estas estructuras con carga dinámicas existen los programas de análisis de elementos finitos como el Abaqus o el NASTRAN/PATRAN que requieren de un complejo uso y que no compensan a la hora de utilizarse con estructuras sencillas.

Debido a la ausencia de un programa que, de forma sencilla, calcule estructura básicas con dichas cargas dinámicas, mi compañero Alfonso Gago Rodríguez, llevo a cabo la creación de un programa que cumpliera dichas expectativas, el "Universal Calculus base don Matrix Method (UCMM)" .

En este proyecto fin de carrera se llevará a cabo una serie de mejoras para dicho programa, generando el UCMMv2.0, y que se irán viendo al detalle más adelante.

1.2. Objetivos

Siguiendo el patrón de trabajo del creador original de dicho programa, es decir, la programación en MatLab de una interfaz de fácil uso, que evite la redundancia de inserción de datos y calcule reacciones y desplazamientos con cargas dinámicas y estáticas se llevarán a cabo las siguientes mejoras:

- La posibilidad de introducir cargas distribuidas y puntuales en cualquier punto de la estructura.
- La posibilidad de la representación de los momentos flectores que generan dichas cargas.
- La posibilidad de representar la deformada que se genera en la estructura.
- La inserción de todas las mejoras citadas anteriormente tanto para estructuras articuladas como reticuladas.

Siendo necesario para el desarrollo de dichos objetivos:

- Un considerable aprendizaje del software de programación MatLab y el desarrollo de su interfaz.

- Desarrollo de un algoritmo que resuelva momentos flectores y deformadas en estructuras articuladas y reticuladas para su posterior representación.

- Familiarización en el cálculo de estructuras con el programa Ed-Tridim.

1.3. Resumen

En el siguiente documento, encontraremos un total de nueve capítulos, en los que encontraremos:

- Este capítulo I en el que se explicaron los objetivos y motivación en la que nos basaremos para la realización de este proyecto.
- Capítulo II en el que se explican los antecedentes teóricos de los que nos hemos ayudado para realizar los cálculos y la programación.
- Capítulo III en el que se realiza una descripción del programa UCMM.
- Capítulo IV en el que se realiza una descripción paso a paso de las mejoras que componen el UCMMv2.0.
- Capítulo V en el que se ha procedido a la validación del UCMMv2.0.
- Capítulo VI en el que se muestran las conclusiones obtenidas sobre el proyecto.
- Capítulo VII en el que se mencionan posibles mejoras futuras.
- Capítulo VIII que forma la bibliografía utilizada para el desarrollo del documento.
- Capítulo IX en el que se adjunta la programación en MatLab que conforma el programa.

2. ANTECEDENTES

2.1. Teoría del cálculo matricial

El programa estará basado en el cálculo matricial, por lo que procederemos a la explicación del método en cuestión.

2.1.1. Método de la rigidez

2.1.1.1. Introducción

Los métodos clásicos de análisis estructural desarrollado a fines del siglo XIX, tienen las cualidades de la generalidad, simplicidad lógica y elegancia matemática.

Desgraciadamente, conducían a menudo a cálculos muy laboriosos cuando se los aplicaba en casos prácticos, y en aquella época, esto era un gran defecto.

Por esta razón sucesivas generaciones de ingenieros se dedicaron a tratar de reducir el conjunto de cálculos. Muchas técnicas ingeniosas de gran valor práctico fueron apareciendo (Método de Cross), pero la mayoría de las mismas eran aplicable sólo a determinados tipos de estructuras.

La principal objeción a los primeros métodos de análisis fue que los mismos conducían a sistemas con un gran número de ecuaciones lineales, difíciles de resolver manualmente.

Con los computadores, capaces de realizar el trabajo numérico, esta objeción no tiene ahora sentido, mientras que la generalidad de los métodos permanece. Esto explica por qué los métodos matriciales deben en su tratamiento básico de las estructuras más al siglo XIX que al XX.

El empleo de la notación matricial presenta dos ventajas en el cálculo de estructuras.
Desde el punto de vista teórico, permite utilizar métodos de cálculo en forma compacta, precisa y, al mismo tiempo, completamente general. Esto facilita el tratamiento de la teoría de estructuras como unidad, sin que los principios fundamentales se vean oscurecidos por operaciones de cálculo, por un lado, o diferencias físicas entre estructuras, por otro.

Desde el punto de vista práctico, proporciona un sistema apropiado de análisis de estructuras y determina una base muy conveniente para el desarrollo de programas de computación.

En contraste con estas ventajas, debe admitirse que los métodos matriciales se caracterizan por una gran cantidad de cálculo sistemático

Las virtudes del cálculo con computadora radican en la eliminación del la preocupación por las operaciones rutinarias, el ingenio necesario para preparar el modelo con que se pretende representar la realidad y el análisis crítico de los resultados.
Se debe ser consciente que sin un modelo adecuado o sin una interpretación final, el refinamiento en el análisis carece de sentido.

2.1.1.2. Método de la Rigidez

Este método se basa en la hipótesis de que:

- Partimos de una estructura lineal, en la que todos los movimientos y esfuerzos son funciones lineales de las cargas
- Las barras son rectas y de sección constante
- Como en cualquier problema estático, se deben cumplir las siguientes ecuaciones:
    - Ecuaciones de compatibilidad
    - Ecuaciones constitutivas
    - Ecuaciones de equilibrio

Las Ecuaciones de compatibilidad relacionan las deformaciones de barras con los desplazamientos nodales. Si se introducen estas relaciones en las Ecuaciones constitutivas, se relacionan las fuerzas en los extremos de barras con los desplazamientos nodales.

Introduciéndose estas últimas relaciones en las Ecuaciones de equilibrio se obtiene un conjunto de ecuaciones de fuerzas nodales en función de desplazamientos nodales, que pueden ser consideradas como Ecuaciones de Equilibrio de la estructura en función de desplazamientos.

La resolución de este sistema de ecuaciones permite obtener el valor de las incógnitas (desplazamientos nodales), a partir de los cuales se obtienen las solicitaciones de las barras de la estructura, así como las reacciones.

Cuando se van a calcular las relaciones esfuerzos de extremos de barra - desplazamientos, es neutral escoger un sistema de coordenadas que haga estas ecuaciones lo más sencillas posible.

Se tomara por lo tanto como eje ‘x’ el que coincide con el eje geométrico de la pieza y los ejes ‘y’ y ‘z’ coincidentes con los ejes principales de la sección transversal.

Tal sistema pertenece a la barra, y no depende de la orientación de la misma en la estructura y se denominara sistema de ejes locales.

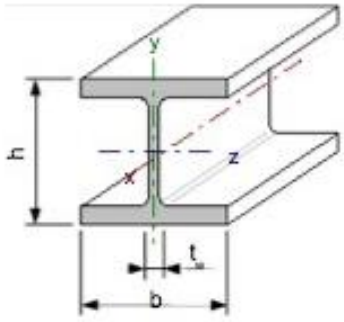


Figura 2. Representación ejes locales

Por el contrario, cuando las piezas se unen entre sí para formar la estructura,

es necesario tener un sistema de coordenadas común para todos los movimientos y esfuerzos de extremo de barras para poder aplicar las condiciones de equilibrio y compatibilidad.

A dicho sistema se lo denominará como sistema de ejes globales.

Por otro lado, dependiendo de la estructura que estemos tratando, tendremos unos desplazamientos u otros:

- Estructura articulada: Dos desplazamientos por nudo, el vertical y el horizontal.
- Estructura reticulada: Tres desplazamientos, los dos citados anteriormente más el giro que se produce en el eje ortogonal a los anteriores.

2.1.1.3. Procedimiento en el análisis matricial de estructuras

A la hora de realizar el análisis de una estructura por el método matricial, podemos distinguir las siguientes partes:

- Identificación estructural
- Cálculo de la matriz de rigidez de barra
- Cálculo cargas nodales
- Rotación de ejes en el plano
- Cálculo de la matriz de rigidez global de la estructura

- Cálculo de la matriz de cargas globales

- Establecer las condiciones de contorno y su consiguiente cálculo de reacciones.

Explicaremos detalladamente cada una de las mismas.

2.1.1.3.1. Identificación estructural

En esta primera parte se definirá la estructura a base de datos y números.

- En primer lugar se definen unos ejes globales para toda la estructura.
- Conectividad de los elementos, se identifica para cada barra el nodo inicial y final. La misma queda definida automáticamente por el orden establecido para la numeración de los nodos de la barra.
  El eje 'x' local coincide con el eje geométrico de la barra, siendo el sentido positivo el que va del nodo de menor numeración al de mayor numeración. Los otros ejes formaran un triedro directo.

2.1.1.3.2. Matriz de rigidez

- Estructura articulada: Consideramos la barra de una estructura articulada con ejes locales x' e y' orientados de la manera que comentábamos anteriormente.

  Supondremos que tratamos con una barra recta de sección transversal constante que responde a la Ley de Hooke.

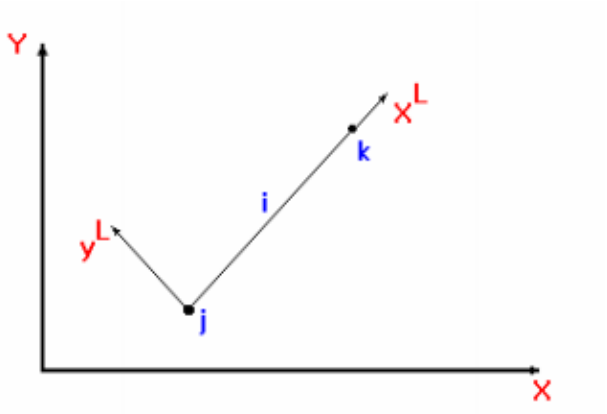


Figura 3. Ejes globales y locales barra

En la barra i de la figura 3, el nodo inicial es el 'j' y el final es el 'k', quedando definida la orientación de la misma por los ejes locales 'x' e 'y'.
Se considera que no existen deformaciones iniciales y que la deformación es elástica. En este caso el alargamiento de la barra 'i' estará dado por:

$$\Delta L_i = X^L_{X_k} - X^L_{X_j}$$

Donde $X^L_{X_k}$ y $X^L_{X_j}$ son los desplazamientos producidos en la barra en dirección del eje local x'

En el caso de estructuras articuladas, la única solicitación que podremos encontrar será el esfuerzo axil, por lo que, siendo el nudo 'j' el inicial y 'k' el final tendremos:

$$F_{X_k} = \frac{E * A}{L} \Delta L_i = \frac{E * A}{L} (X^L_{X_k} - X^L_{X_j})$$

$$F_{X_j} = \frac{E * A}{L} \Delta L_i = -\frac{E * A}{L} (X^L_{X_k} - X^L_{X_j})$$

Siendo:

E, El módulo de elasticidad, A el área transversal y L la longitud de la barra.

Sabiendo que en el eje y' local de cada barra no podemos encontrar ningún esfuerzo, podremos expresar de forma matricial las ecuaciones anteriores:

$$\begin{bmatrix} F_{X_j} \\ F_{Y_j} \\ F_{X_k} \\ F_{Y_k} \end{bmatrix} = \begin{bmatrix} +\frac{E * A}{L} & 0 & -\frac{E * A}{L} & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{E * A}{L} & 0 & +\frac{E * A}{L} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} X^L_{X_j} \\ X^L_{Y_j} \\ X^L_{X_k} \\ X^L_{Y_k} \end{bmatrix}$$

La expresión anterior se corresponde a la ecuación matricial de la barra 'i' en coordenadas locales y expresa las fuerzas de extremo de barra FI^L en función de los desplazamientos de los nodos XI^L.

A la matriz que relaciona FI^L y XI^L se la denomina matriz de rigidez de barra de la estructura articulada en coordenadas locales KI^L.

Expresado en forma compacta o simbólica:

$$FI^L = KI^L * XI^L \qquad \textbf{(1)}$$

En esta ecuación vienen definidas las fuerzas extremo de ambos nodos, el inicial y el final de la barra para cualquier pareja de desplazamientos X_j , X_k . Estas ecuaciones son simétricas, como se podía esperar a partir del teorema de reciprocidad. No es posible, sin embargo, resolverlas y obtener los desplazamientos (X) en términos de las fuerzas (F), puesto que la matriz K es singular. Esta información nos lleva a concluir que las barras pueden estar sometidas a cualquier esfuerzo arbitrario que no afectará a los movimientos del nodo inicial y final de la barra.

En el caso de que la estructura esté dispuesta de un muelle sometido a tracción.

$$F_{X_k} = K * \Delta L_i = K * (X^L_{X_k} - X^L_{X_j})$$

(2)

$$F_{Yk} = K * \Delta L_i = K * (X^L_{X_k} - X^L_{X_j})$$

(3)

Donde K es la constante de rigidez del muelle.

$$\begin{bmatrix} F_{X_j} \\ F_{Y_j} \\ F_{X_k} \\ F_{Y_k} \end{bmatrix} = \begin{bmatrix} K & 0 & K & 0 \\ 0 & 0 & 0 & 0 \\ K & 0 & K & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} X^L_{X_j} \\ X^L_{Y_j} \\ X^L_{X_k} \\ X^L_{Y_k} \end{bmatrix}$$

(4)

Mientras que si nos encontramos con un muelle que haga de efecto disipador tendremos para los nudos ‘j’ y ‘k’.

$$F_{X_k} = C * \Delta L_i = C * (\dot{X}^L_{X_k} - \dot{X}^L_{X_j})$$

(5)

$$F_{Y_k} = C * \Delta L_i = C * (\dot{X}^L_{X_k} - \dot{X}^L_{X_j})$$

(6)

Siendo C la constante de amortiguamiento del amortiguador.

$$\begin{bmatrix} F_{X_j} \\ F_{Y_j} \\ F_{X_k} \\ F_{Y_k} \end{bmatrix} = \begin{bmatrix} C & 0 & C & 0 \\ 0 & 0 & 0 & 0 \\ C & 0 & C & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} \dot{X}^L_{X_j} \\ \dot{X}^L_{Y_j} \\ \dot{X}^L_{X_k} \\ \dot{X}^L_{Y_k} \end{bmatrix}$$

(7)

- Estructura reticulada: En este tipo de estructura nos encontramos con los movimientos traslacionales en el eje horizontal y vertical más el giro en el plano.

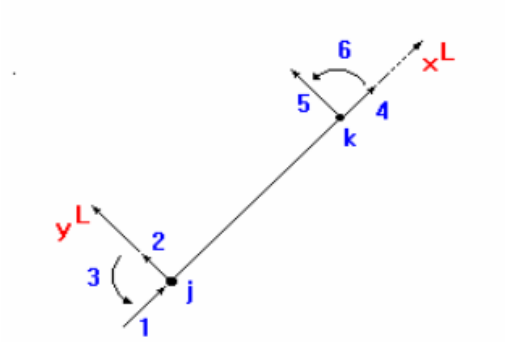


Figura 4. Barra reticulada.

Para la obtención de la matriz de rigidez, se aplican desplazamientos de valor unitario en uno de los movimientos disponibles y se restringen los demás.

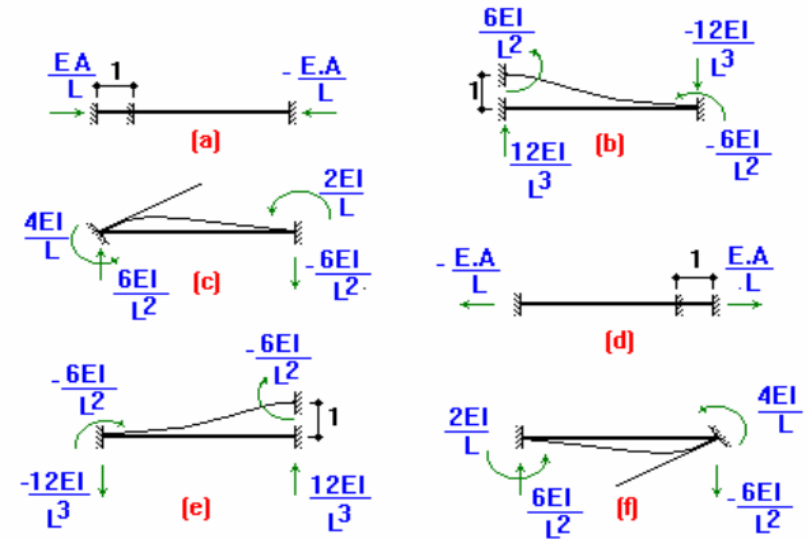


Figura 5. Reacciones unitarias

Las reacciones que se hallan con los desplazamientos unitarios serán los términos que formarán la matriz de rigidez de la estructura:

$$\begin{bmatrix} F_{Xj} \\ F_{Yj} \\ F_{Zj} \\ F_{Xk} \\ F_{Yk} \\ F_{Zk} \end{bmatrix} = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix} * \begin{bmatrix} X_{Xj}^L \\ X_{Yj}^L \\ X_{Zj}^L \\ X_{Xk}^L \\ X_{Yk}^L \\ X_{Zk}^L \end{bmatrix}$$

(8)

De la misma forma que ocurría en estructuras articuladas, expresada de forma compacta:

$$FI^L = KI^L * XI^L$$

(9)

Para un muelle:

$$\begin{bmatrix} F_{Xj} \\ F_{Yj} \\ F_{Zj} \\ F_{Xk} \\ F_{Yk} \\ F_{Zk} \end{bmatrix} = \begin{bmatrix} K & 0 & 0 & K & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ K & 0 & 0 & K & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} X_{Xj}^L \\ X_{Yj}^L \\ X_{Zj}^L \\ X_{Xk}^L \\ X_{Yk}^L \\ X_{Zk}^L \end{bmatrix}$$

(10)

Para un amortiguador:

$$\begin{bmatrix} F_{Xj} \\ F_{Yj} \\ F_{Zj} \\ F_{Xk} \\ F_{Yk} \\ F_{Zk} \end{bmatrix} = \begin{bmatrix} C & 0 & 0 & C & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ C & 0 & 0 & C & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} \dot{X}_{Xj}^L \\ \dot{X}_{Yj}^L \\ \dot{X}_{Zj}^L \\ \dot{X}_{Xk}^L \\ \dot{X}_{Yk}^L \\ \dot{X}_{Zk}^L \end{bmatrix}$$

(11)

2.1.1.3.3. Vector de cargas nodales

Hasta ahora se ha supuesto que las cargas estaban aplicadas en los nodos, y por lo tanto existe una correspondencia biunívoca entre los puntos de aplicación de las cargas y los desplazamientos que están siendo calculados. Si esto no ocurriera, por ejemplo se tuviesen cargas en el tramo de las barras, en forma distribuida o concentrada, se debe sustituir las cargas en las mismas por un sistema de cargas equivalentes aplicadas en los nodos que produzca en la estructura el mismo efecto que las cargas originales.

Aplicando el principio de superposición, que es válido por haber supuesto que el sistema es lineal, se puede descomponer las cargas tal como se indica en la figura:

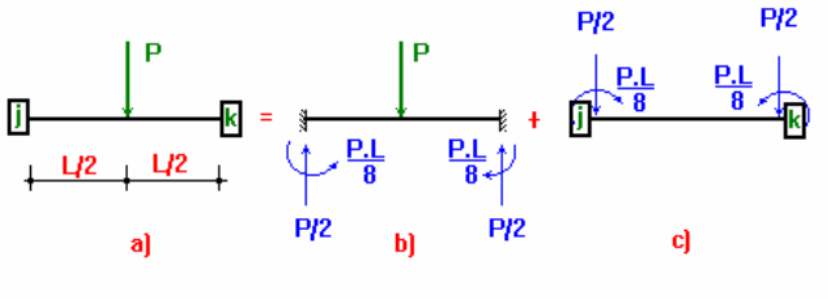


Figura 6. Cargas en los nodos.

Como se puede observar las cargas, reacciones y deformaciones de la estructura a) serán equivalentes a la suma de los dos estados b) y c).

Como las deformaciones de nodos en b) son nulas, serán iguales las deformaciones de los casos c) y a). O sea que las cargas de c) producen la misma respuesta estructural en lo referente a desplazamientos de nudos que las cargas originales.

Estas serán entonces las cargas equivalentes en los nodos, que serán las reacciones que se producen en el empotramiento perfecto pero cambiadas de signo.

Los esfuerzos en los extremos de barra se obtienen por la suma de los casos (b) y (c).

$$F^a = F^b + F^c D^a = D^c \quad (12)$$

Por lo tanto, a la ecuación compacta habrá que añadirle las fuerzas del empotramiento perfecto, el caso (b).

$$FI^L = KI^L * XI^L + AI^L \quad (13)$$

Donde AI^L representa el vector de fuerzas del empotramiento perfecto en ejes locales.

2.1.1.3.4. Rotación de ejes en el plano

Hasta el momento, las matrices de rigidez elaboradas, tanto para estructuras articuladas como reticuladas, están referenciadas a los ejes globales de cada barra, siendo el desplazamiento y las fuerzas referidas a los nodos inicial y final de las mismas.

Para llevar a cabo las operaciones que nos resuelvan la estructura deseada, debemos transformas estas matrices en ejes locales de cada barra a un sistema global para toda la estructura.

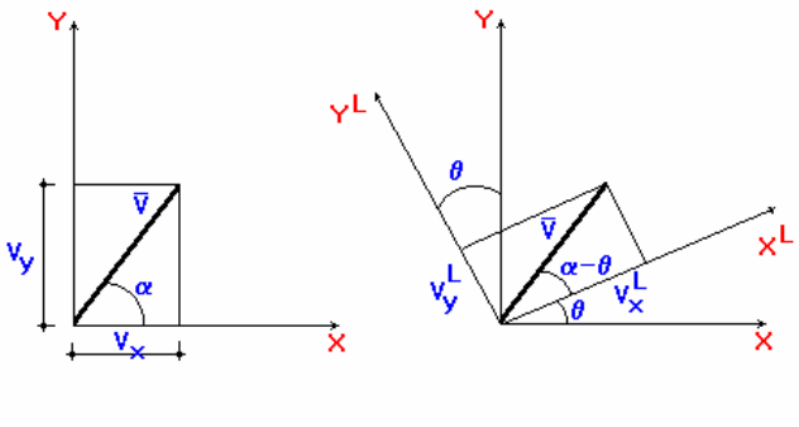


Figura 7. Ángulo entre ejes globales y locales.

Suponiendo el vector V a los ejes 'x' e 'y' sus componentes serán:

$$Vx = V * cos\,\alpha \tag{14}$$

$$Vy = V * sen\,\alpha \tag{15}$$

Mientras que en función de los ejes locales de la barra X^L e Y^L :

$$Vx^L = V * cos(\alpha - \theta) \tag{16}$$

$$Vy^L = V * sen(\alpha - \theta) \tag{17}$$

Con lo que obtenemos:

$$Vx^L = V * cos\,\alpha * cos\theta + V * sen\alpha * sen\theta \tag{18}$$

$$Vy^L = V * cos\,\alpha * cos\theta + V * sen\alpha * sen\theta \tag{19}$$

Lo cual expresándolo en forma matricial concluimos en:

$$\begin{bmatrix} Vx^L \\ Vy^L \end{bmatrix} = \begin{bmatrix} cos\theta & sen\theta \\ -sen\theta & cos\theta \end{bmatrix} * \begin{bmatrix} Vx \\ Vy \end{bmatrix} \quad \textbf{(20)}$$

De forma compacta:

$$V^L = R * V \quad \textbf{(21)}$$

A la matriz R de senos y cosenos se le llamará matriz de rotación, y será la que nos transforme los términos que tengamos en locales a globales para asi poder operar con ellos. Para determinar el ángulo será el resultante de girar el eje X^G en sentido positivo hacia el X^L.

Esta matriz de 2x2 se utilizará para transformar estructuras articuladas mediante la siguiente ecuación:

$$[K^G] = [R] * [K^L] * [R^T] \quad \textbf{(22)}$$

Siendo:
K^G la matriz de rigidez en ejes globales .
K^L la matriz de rigidez en ejes locales.
R y R^T la matriz de rotación vista anteriormente y su traspuesta.

En el caso de encontrarnos con una estructura reticulada, la matriz de rotación R será:

$$[R] = \begin{bmatrix} cos\theta & -sen\theta & 0 \\ sen\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{(23)}$$

2.1.1.3.5. Matriz global de rigidez de la estructura

La matrices de rigidez y amortiguamiento global de la estructura tendrá tantas filas y tantas columnas como nodos por sus grados de libertad tenga la estructura.

Se obtiene de calcular las submatrices de cada barra en la posición correspondiente. En las posiciones en las que se tenga varias submatrices se producirá el ensamble de la matriz, y a ello lo llamaremos elementos ensamblados.

Veamos cómo funciona apoyándonos en la siguiente estructura:

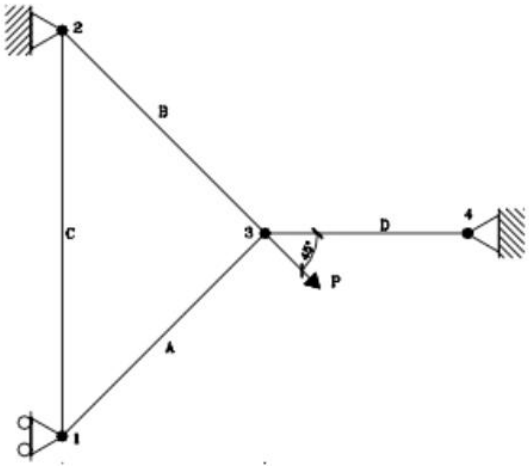


Figura 8. Estructura articulada.

Tenemos cuatro barras:

Barra A, cuyo nodo menor es el 1 y el nodo mayor es el 3
Barra B, cuyo nodo menor es el 2 y el nodo mayor es el 3
Barra C, cuyo nodo menor es el 1 y el nodo mayor es el 2
Barra D, cuyo nodo menor es el 3 y el nodo mayor es el 4
Cada barra tendrá una matriz de rigidez que ocupará cierto lugar en la matriz global.

Barra A

$$\begin{bmatrix} (K_{11}^{A})_{11} & (K_{12}^{A})_{13} \\ (K_{21}^{A})_{31} & (K_{22}^{A})_{33} \end{bmatrix}$$

Barra C

$$\begin{bmatrix} (K_{11}^{C})_{11} & (K_{12}^{C})_{12} \\ (K_{21}^{C})_{21} & (K_{22}^{C})_{22} \end{bmatrix}$$

Barra B

$$\begin{bmatrix} (K_{11}^{B})_{22} & (K_{12}^{B})_{23} \\ (K_{21}^{B})_{32} & (K_{22}^{B})_{33} \end{bmatrix}$$

Barra D

$$\begin{bmatrix} (K_{11}^{D})_{33} & (K_{12}^{D})_{34} \\ (K_{21}^{D})_{43} & (K_{22}^{D})_{44} \end{bmatrix}$$

Cada una de estas submatrices, irán ubicadas en la matriz de rigidez global de la estructura según la posición que marca el subíndice que aparece fuera del paréntesis. Aquellas submatrices que tengan el mismo subíndice, se sumaran en la matriz de rigidez de la estructura.
Obteniéndose la siguiente matriz de rigidez de la estructura, en este caso en ejes locales:

$$\begin{bmatrix} \left(K_{11}^{A}\right)_{11}+\left(K_{11}^{C}\right)_{11} & \left(K_{12}^{C}\right)_{12} & \left(K_{12}^{A}\right)_{13} & 0 \\ \left(K_{21}^{C}\right)_{21} & \left(K_{11}^{B}\right)_{22}+\left(K_{22}^{C}\right)_{22} & \left(K_{12}^{B}\right)_{23} & 0 \\ \left(K_{21}^{A}\right)_{31} & \left(K_{21}^{B}\right)_{32} & \left(K_{22}^{A}\right)_{33}+\left(K_{22}^{B}\right)_{33}+\left(K_{11}^{D}\right)_{33} & \left(K_{12}^{D}\right)_{34} \\ 0 & 0 & \left(K_{21}^{D}\right)_{43} & \left(K_{22}^{D}\right)_{44} \end{bmatrix}$$

2.1.1.3.6. Matriz de cargas globales de la estructura

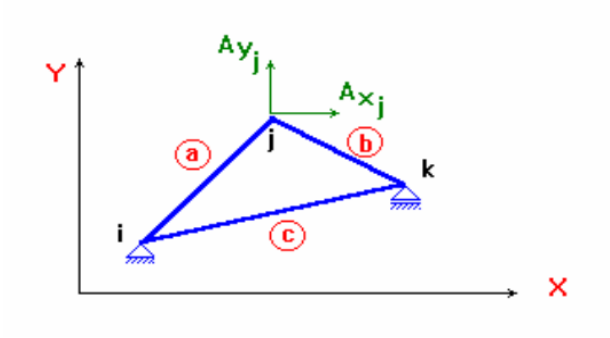


Figura 9. Ejemplo estructura.

La ecuaciones de equilibrio exigen que las cargas externas aplicadas en los nodos deben ser iguales a la suma de las solicitaciones de extremo de las barras que concurren al nodo.

Siendo el vector de cargas externas aplicadas en j:

$$Aj = \begin{bmatrix} A_{xj} \\ A_{yj} \end{bmatrix} \quad \textbf{(24)}$$

Que centrándonos únicamente en la barra (a) obtenemos la siguiente ecuación matricial:

$$F^a = K^a * X^a + A^a \;\rightarrow\; F^a - A^a = K^a * X^a \quad \textbf{(25)}$$

Donde:
F^a : Solicitación en el extremo 'j' de la barra (a)
X^a : Deformación en el extremo 'j' de la barra (a)
A^a : Fuerza de empotramiento perfecto en el extremo 'j' de la barra (a)
K^a: Matriz de rigidez en la barra (a)

2.1.1.3.7. Condiciones de contorno y cálculo de reacciones

Un sistema de ecuaciones: A = K · X correspondiente a una estructura completa antes de aplicarse las condiciones de contorno es indeterminado, pues K es singular.
La razón de esta singularidad es el resultado de no haberse considerado las vinculaciones o apoyos de la estructura en el exterior. Al introducirse las condiciones de vínculo desaparece la indeterminación, siempre que el número de vínculos sea por lo menos el mínimo necesario para eliminar los movimientos de cuerpo rígido de la estructura.

El conocimiento de determinados movimientos nodales, disminuye el número de incógnitas, tornándose innecesarias las ecuaciones correspondientes a estos corrimientos.

La eliminación de la ecuación de un desplazamiento implica la destrucción de la banda de la matriz, lo cual exigirá un reacomodamiento de las incógnitas del problema.

Se puede expresar el sistema de ecuaciones A = K · X, de la siguiente forma:

$$\begin{bmatrix} A_H \\ A_V \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} * \begin{bmatrix} 0 \\ X_V \end{bmatrix} \quad \textbf{(26)}$$

Siendo:
A_H : Reacciones de apoyo con desplazamiento impedido (XH = 0), reacciones que por el momento se desconocen.

A_V : Agrupa las fuerzas externas sobre nodos con desplazamientos XV desconocidos.

K: matriz de rigidez que relaciona fuerzas conocidas (cargas externas A V) con desplazamientos desconocidos XV.
Por lo que el sistema de ecuaciones reducido que tenemos que plantear finalmente es:

$$A_V = K_{22} * X_V \quad \textbf{(27)}$$

Una vez que obtenemos las incógnitas de los desplazamientos, podemos proceder con el cálculo de las reacciones por medio de la siguiente ecuación de matrices.

$$F - A = K * X \quad \textbf{(28)}$$

Que nos proporcionará el vector F, resultado de las reacciones de la estructura.

2.2. Deformación en vigas

2.2.1. Introducción

El análisis estructural de las vigas suele dividirse en vigas isostáticas e hiperestáticas. Recordemos que esta división corresponde a las condiciones de apoyo que presente el elemento a analizar Si la viga tiene un número igual o inferior a tres incógnitas en sus reacciones, bastará con aplicar las condiciones de equilibrio estático para resolverla.

$$\sum Fx = 0 \qquad \sum Fy = 0 \qquad \sum M = 0$$

Si en cambio, la viga presenta un mayor número de incógnitas, no bastará con las ecuaciones antes indicadas, sino que será necesario incorporar nuevas expresiones.

Para abordar el análisis de las vigas hiperestáticas o estáticamente indeterminadas resulta necesario analizar las deformaciones que experimentará la viga, luego de ser cargada. Las distintas cargas sobre la viga generan tensiones de corte y flexión en la barra, y a su vez la hacen deformarse.

El análisis de las deformaciones tiene básicamente dos objetivos. Por una parte, el poder obtener nuevas condiciones, que traducidas en ecuaciones, nos permitan resolver las incógnitas en vigas hiperestáticas. Y por otra parte, las deformaciones en sí, deben ser limitadas. Los envigados de madera o acero, por ejemplo, pueden quedar correctamente diseñados por resistencia, vale decir, no se romperán bajo la carga, pero podrán deformarse más allá de lo deseable, lo que llevaría consigo el colapso de elementos de terminación como cielos falsos o ventanales. No resulta extraño entonces que muchos dimensionamientos queden determinados por la deformación y no por la resistencia.

2.2.2. Línea elástica

Denominaremos línea elástica a la curva que forma la fibra neutra una vez cargada la viga, considerando que ésta se encontraba inicialmente recta.

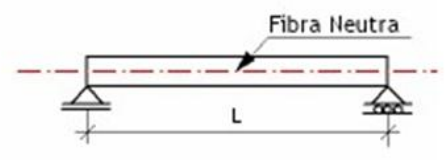


Figura 10. Fibra neutra.

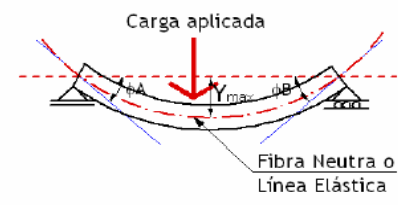


Figura 11. Fibra neutra [2].

2.2.3. Supuesto base

Para establecer una serie de relaciones al interior de la sección, indicamos que se trata de una viga, cuyo material se encuentra solicitado dentro del rango de proporcionalidad entre tensiones y deformaciones, y en donde se admite la conservación de las caras planas. Dicho en otra forma, donde se cumplen la ley de Hooke y la hipótesis de Bernouilli-Navier.

2.2.3.1. Ley de Hooke

Establece que la relación entre la tensión y la deformación unitaria es una constante y se denomina módulo de elasticidad.

$$E = \frac{\tau}{e} \quad \rightarrow \quad \tau = E * e$$

(29)

Siendo:

E: Elasticidad

e: La deformación unitaria

τ: Tensión

2.2.3.2. Deducción de la formula de flexión

De la deducción realizada para dimensionar elementos sometidos a la flexión simple sabemos que:

$$\tau = \frac{M * V}{I}$$

(30)

Siendo:

M: Momento Flector.

V: La distancia desde la fibra neutra a la parte más comprimida o traccionada.

I: El momento de Inercia.

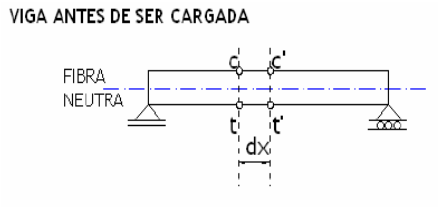
Si combinamos ambas expresiones de la tensión obtenemos:

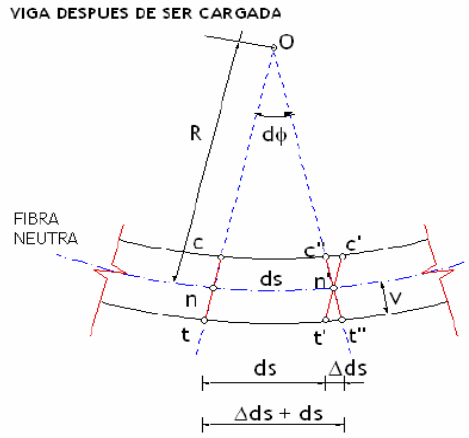
$$E * e = \frac{M * V}{I} \quad \rightarrow \quad e = \frac{M * V}{IE}$$

(31)

2.2.3.3. Análisis de la sección

La sección cc'tt', inicialmente recta, se curva con un radio R como indica el gráfico.





La fibra cc' se acorta a cc'', la fibra tt' se alarga a tt'' y la nn' permanece constante. Por triángulos semejantes non' y t'n't'' obtenemos

$$\frac{\Delta ds}{ds} = \frac{V}{R} = \varepsilon \text{ (Deformación unitaria)}$$

$$\frac{I}{R} = \frac{d\emptyset}{ds} \quad \rightarrow \frac{I}{R} = \frac{M}{EI} = \frac{d\emptyset}{ds}$$

$$d\emptyset = \frac{M\ ds}{EI}$$

(32)

Como nos estamos refiriendo a una sección infinitamente pequeña, la diferencia entre un arco y su proyección horizontal es mínima: ds ≈ dx.

La expresión final indica que la curvatura de la línea elástica es una variable proporcional al momento flector.

$$d\emptyset = \frac{M * dx}{EI}$$

(33)

2.2.4. Métodos de cálculo

Existen diferentes métodos para abordar el análisis de las deformaciones en las vigas:

- Método de Área de Momentos.
- Método de Doble Integración.
- Método de la Viga Conjugada.

Si bien, todos presentan su mecánica propia, a la vez tienen una partida común, que es justamente el análisis de la elástica expuesto anteriormente.

En este caso, para la resolución de los problemas planteados, profundizaremos en el método de la doble integración.

2.2.4.1. Método de la doble integración

Partiendo del análisis anterior, en el cual deducíamos la ecuación de la elástica, y con la misma premisa de que trabajaremos con ángulos bastante pequeños:

$$d\emptyset = \frac{M * dx}{EI}$$

$$tan\,\emptyset = \emptyset$$

(34)

La derivada en cualquier punto de la una curva es igual a la pendiente de la tangente a la curva en ese punto.

$$tan\,\emptyset = \emptyset = \frac{dy}{dx}$$

(35)

Reemplazando en la ecuación inicial obtenemos la Ecuación Diferencial de la Elástica de una viga:

$$\frac{d^2y}{dx} = \frac{M}{EI}$$

(36)

Ecuación que al integrarla obtenemos la Ecuación general de la pendiente:

$$\frac{dy}{dx} = \frac{1}{EI}\int M\,dx + C_1$$

(37)

Si la ecuación anterior, la integramos de nuevo, hallamos la Ecuación general de la flecha:

$$y = \frac{1}{EI}\iint M\,dx + C_1 + C_2$$

(38)

Este método nos permite calcular las pendientes y deflexiones de la viga en cualquier punto. La dificultad radica en despejar las constantes de integración. Esto se logra analizando las condiciones de apoyo y la deformación de la viga.

2.3. MatLab

2.3.1. Introducción

MATLAB es el nombre abreviado de "MATrix LABoratory". MATLAB es un programa para realizar cálculos numéricos con vectores y matrices. Como caso particular puede también trabajar con números escalares -tanto reales como complejos-, con cadenas de caracteres y con otras estructuras de información más complejas. Una de las capacidades más atractivas es la de realizar una amplia variedad de gráficos en dos y tres dimensiones. MATLAB tiene también un lenguaje de programación propio.

2.3.2. Entorno gráfico de MatLab

Se trata de un entorno de trabajo muy gráfico e intuitivo , cuyas componentes mas importante son:

1. El escritorio de MatLab (MatLab Desktop), que es la ventana o contenedor de máximo nivel en la que se pueden situar los demás componentes.

2. Las componentes individuales, que podrán ser colocadas o no, a gusto del usuario, en el escritorio de MatLab, orientadas a tareas concretas dentro del programa, entre las que se puede citar:

a. La ventana de comandos (Command Window).

b. La ventana histórica de comandos (Command History).

c. El espacio de trabajo (Workspace).

d. La plataforma de lanzamiento (Launch Pad).

e. El directorio actual (Current Folder).

f. La ventana de ayuda (Help)

g. El editor de ficheros y depurador de errores (Editor).

h. El editor de vectores y matrices (Array Editor).

i. . La ventana que permite estudiar cómo se emplea el tiempo de ejecución (Profiler).

A continuación se proporcionará una breve descripción de cada una de las partes mencionadas.

2.3.2.1. Escritorio de MatLab (MatLab Desktop)

El Matlab Desktop es la ventana más general de la aplicación. El resto de las ventanas o componentes citadas pueden alojarse en la Matlab Desktop o ejecutarse como ventanas independientes. A su vez, los componentes alojados en el Matlab Desktop pueden aparecer como sub-ventanas independientes o como pestañas dentro de una de las sub-ventanas. Cuando se arranca MATLAB por primera vez aparece una imagen como la de la figura 12.

Cada vez que se ejecuta el programa MatLab, se abre el escritorio con la forma en que se cerró, siendo posible almacenar, con distintos nombres, diferentes diseños del escritorio.

En la figura que se muestra a continuación, podemos observar el directorio (Current Folder), la ventana de comandos (Command Window), el espacio de trabajo (Workspace) y el historial de comandos introducidos (Command History).

Esta es la configuración que trae por defecto el software, siendo, como ya hemos comentado, perfectamente editable.

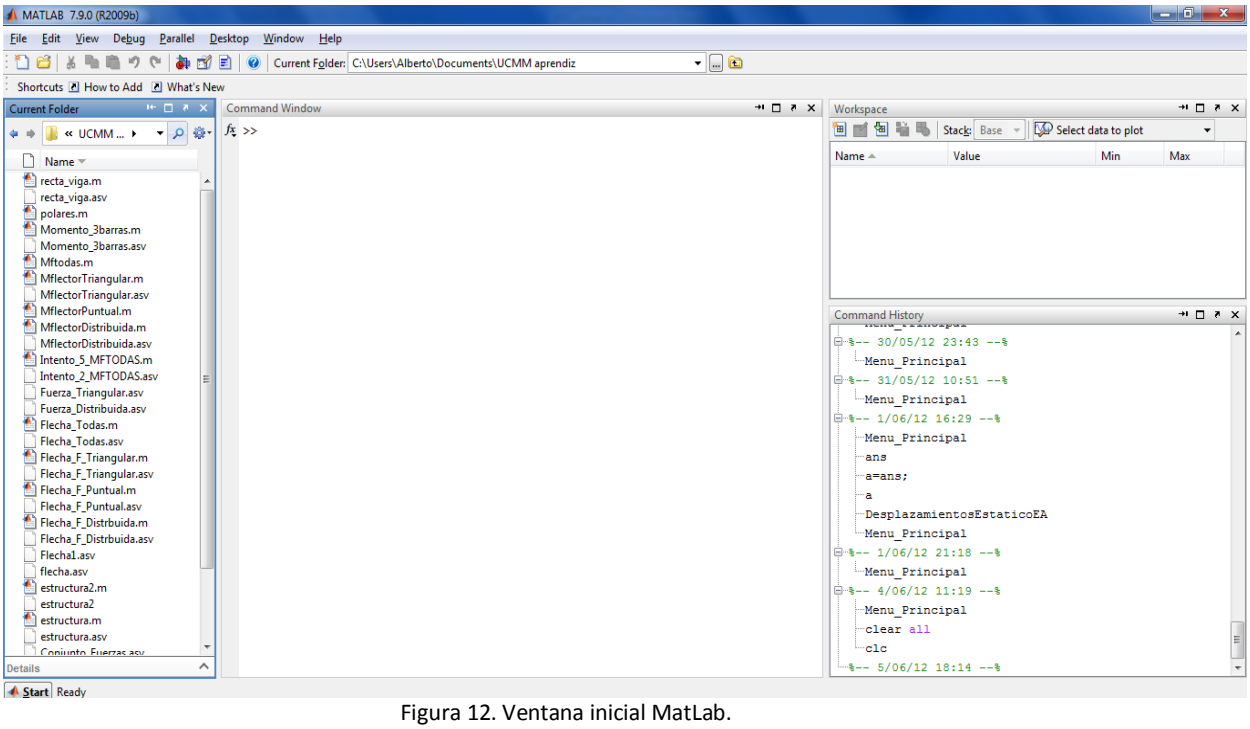


Figura 12. Ventana inicial MatLab.

2.3.2.2. Command Window

Es la parte más importante de la interfaz, en la figura mostrada es la parte más centrada y de mayor tamaño. En ella introducimos todas las operaciones que queremos llevar a cabo en el programa.

Todas las sentencias que vayamos ejecutando en el command window se irán guardando y podrán ser recuperadas en cualquier momento mediante las teclas de la flecha hacia arriba y hacia abajo ↑↓.

Al igual que nos permite introducir datos y sentencias, será la parte donde se muestren los resultados, tanto de los comandos que se ejecuten directamente en la ventana como los que se ejecuten a través de un archivo .m

2.3.2.3. Command History

En esta ventana, situada en la parte inferior izquierda de la imagen, se recogen todas las sentencias que han sido ejecutadas asi como los cierres e incios de sesión que se han producido. Nos permite recuperar dichas sentencias que fueron ejecutadas.

2.3.2.4. Current Folder

Los programas MatLab se encuentran en ficheros con la extensión *.m. Estos ficheros se ejecutan tecleando su nombre en la línea de comandos (sin la extensión), seguido de los argumentos entre paréntesis, si se trata de funciones. No todos los ficheros *.m son accesibles sin más. Para que un fichero *.m pueda ser ejecutado es necesario que se cumpla una de las siguientes condiciones:

1. Que esté en la Current Folder (ventana en la zona izquierda de la Figura mostrada) . MatLab mantiene en todo momento un único directorio con esta condición y es el primer sitio donde MatLab busca cuando desde la línea de comandos se le pide que ejecute un fichero.

2. Que esté en uno de los directorios indicados en el Path de MatLab.
   El Path es una lista ordenada de directorios en los que el programa busca los ficheros o las funciones que ha de ejecutar.

La ventana Current Folder permite explorar los directorios del ordenador, mostrando los ficheros deseados. Los ficheros *.m mostrados en la ventana Current Folder se pueden abrir con el Editor/Debugger mediante un doble clic. A partir del menú contextual que se abre clicando con el botón derecho en cualquier parte de la ventana Current Folder se tiene la posibilidad de añadir ese directorio al Path de MatLab.

2.3.2.5. Workspace y Variable Editor

Estas dos ventanas que nos ofrece el escritorio de MatLab guardan una estrecha relación. El Workspace, parte situada a la derecha de la imagen inferior, es el lugar donde se almacenan todas las variables que han ido apareciendo en las sentencias que ha desarrollado el programa. Estas variables que se han obtenido, pueden ser modificadas en cualquier momento haciendo doble clic en la variable. Este doble clic nos abrirá el Variable Editor, parte de la izquierda de la imagen, donde podremos ver la variable seleccionada, ya sea un escalar, un vector o una matriz.

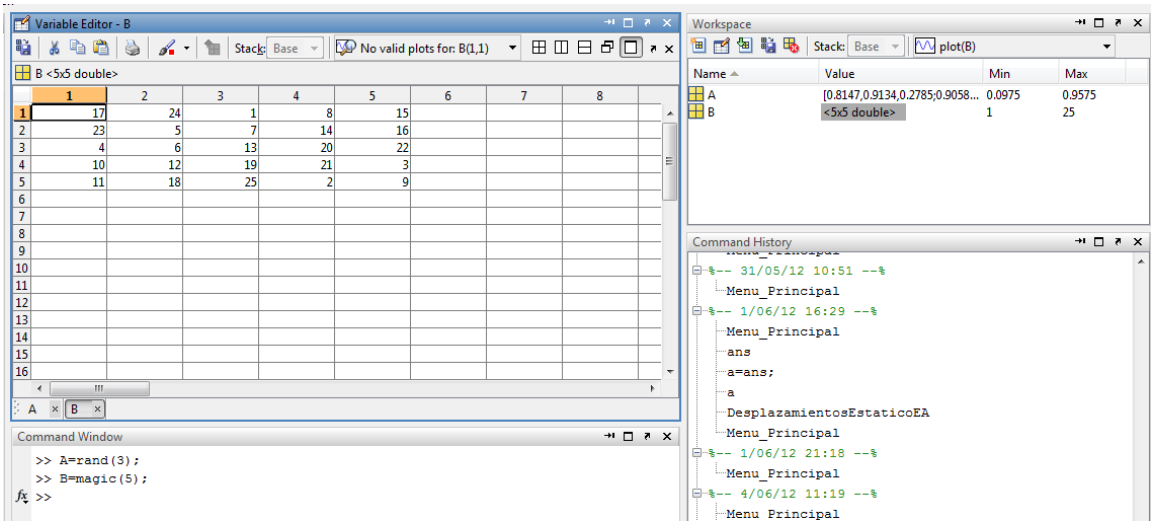


Figura 13. Editor de variables de Matlab.

2.3.2.6. Editor/Debugger

En MatLab tienen particular importancia los ya citados ficheros-M (o M-files). Son ficheros de texto ASCII, con la extensión *.m, que contienen conjuntos de comandos o definición de funciones. La importancia de estos ficheros-M es que al teclear su nombre en la línea de comandos y pulsar Intro, se ejecutan uno tras otro los comandos contenidos en dicho fichero. El poder guardar instrucciones y grandes matrices en un fichero permite ahorrar mucho trabajo de tecleado.

MatLab dispone de un editor que permite tanto crear y modificar estos ficheros, como ejecutarlos paso a paso para ver si contienen errores (proceso de Debug o depuración)

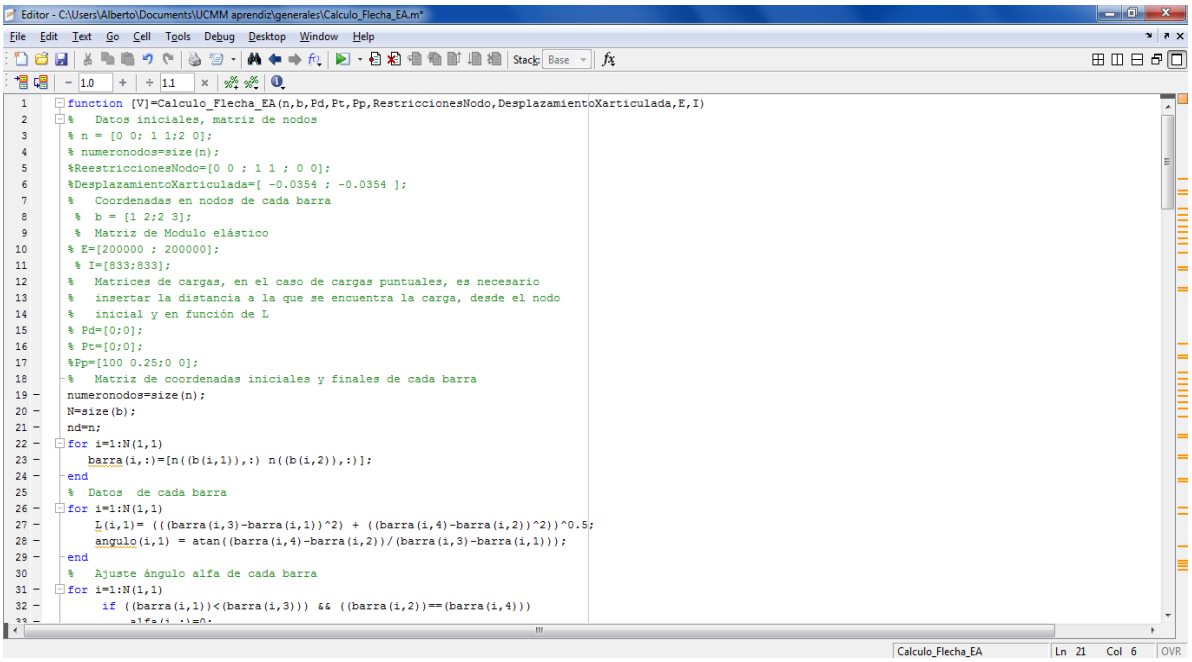


Figura 14. Editor de ficheros.

En el editor aparece el archivo *.m entero para su revisión, cabe destacar que las partes en verde, con el símbolo % al principio de la línea, significa que es texto y que el programa no lo tendrá en cuenta.

El resto, en negro y azul, serán las sentencias que se llevarán a cabo al ejecutar el archivo.

Una vez que hemos comentado las partes más comunes para utilizar la interfaz de MatLab procederemos con la explicación para la creaciones de GUIDES en MatLab.

2.3.3. GUIDE, la interfaz gráfica

GUIDE (Graphical User Interface Development Environment) es un juego de herramientas que se extiende por completo en el soporte Matlab, diseñadas para crear GUIs (Graphical User Interfaces) de formas bastante sencilla e Intuitiva.

MATLAB permite desarrollar de manera simple un conjunto de pantallas con botones, menús, ventanas, etc., que permiten utilizar de manera muy simple programas realizados en entorno. Este conjunto de herramientas se denomina interface de usuario. Las posibilidades que ofrece MATLAB no son muy amplias, en comparación a otras aplicaciones de Windows como Visual Basic. Para poder hacer programas que utilicen las capacidades gráficas avanzadas de MATLAB hay que conocer algunos conceptos que se explican en los apartados siguientes.

2.3.3.1. Estructura de los gráficos en Matlab.

Los gráficos de MATLAB tienen una estructura jerárquica formada por objetos de distintos tipos. Esta jerarquía tiene forma de árbol, con el aspecto mostrado en la figura siguiente.

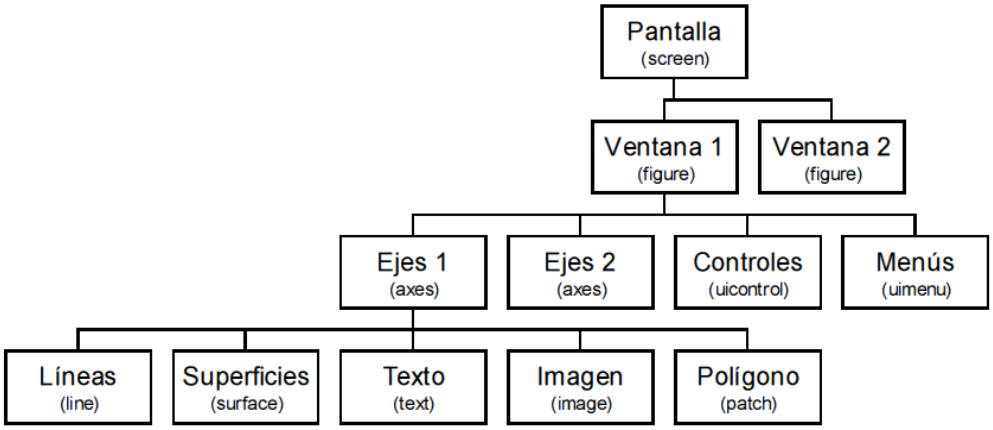


Figura 15. Jerarquía estructural MatLab.

Según se muestra en la figura 15, el objeto más general es la pantalla (screen). Este objeto es la raíz de todos los demás y sólo puede haber un objeto pantalla. Una pantalla puede contener una o más ventanas (figures). A su vez cada una de las ventanas puede tener uno o más ejes de coordenadas (axes) en los que representar otros objetos de más bajo nivel. Una ventana puede tener también controles (uicontrol) tales como botones, barras de desplazamiento, botones de selección o de opción, etc.) y menús (uimenu). Finalmente, los ejes pueden contener los cinco tipos de elementos gráficos que permite MATLAB: líneas (line), polígonos (patches), superficies (surface), imágenes bitmap (image) y texto (text).

La jerarquía de objetos mostrada en la figura 15 indica que en MATLAB hay objetos padres e hijos. Por ejemplo, todos los objetos ventana son hijos de pantalla, y cada ventana es padre de los objetos ejes, controles o menús que están por debajo. A su vez los elementos gráficos (líneas, polígonos, etc.) son hijos de un objeto ejes, y no tienen otros objetos que sean sus hijos.

Cuando se borra un objeto de MATLAB automáticamente se borran todos los objetos que son sus descendientes. Por ejemplo, al borrar unos ejes, se borran todas las líneas y polígonos que son hijos suyos.

Todos los objetos que aparecen en un gráfico tienen un identificador único llamado handle.

2.3.3.2. Propiedades de los objetos

Todos los objetos de MATLAB tienen distintas propiedades. Algunas de éstas son el tipo, el estilo, el padre, los hijos, si es visible o no, y otras propiedades particulares del

objeto concreto de que se trate. Las propiedades comunes a todos los objetos son: children, clipping, parent, type, UserData, Visible.

2.3.3.2.1. Funciones set y get

MATLAB dispone de las funciones set y get para consultar y cambiar el valor de las propiedades de un objeto. Las funciones set(id) lista en pantalla todas las propiedades del objeto al que corresponde el handle (sólo los nombres, sin los valores de las propiedades). La función get(id) produce un listado de las propiedades y de sus valores. En resumen, get almacena y set muestra lo almacenado.

2.3.3.3. Creación de controles gráficos : Comando uicontrol

El formato uicontrol permite definir los controles gráficos para manejar un programa. Algunas de las propiedades del comando uicontrol:

- Color de objeto (BackgroundColor) : controla el color del objeto, por defecto éste será gris claro.

- Acción a efectuar por el comando (CallBack): este comando especifica la acción a efectuar por MatLab al actuar sobre el control.

- Control activado/desactivado (Enable): este comando permite desactivar un control, de tal forma que una acción sobre el mismo (por ejemplo, apretar con el ratón sobre un PushButton) no produce ningún efecto. Los valores que puede tomar esta variable son on y off.

- Alineamiento horizontal de título (HorizontalAlignment): esta opción determina la posición del título del control en el mismo (propiedad String).

- Valor máximo (Max): determina el máximo valor que puede tomar la propiedad Value cuando se utilizan cajas de textos, botones o barras de desplazamiento.

- Valor mínimo (Min): análogo a la opción anterior pero para el valor mínimo.

- Identificador del objeto padre (Tag): esta opción especifica el tag del objeto padre. Uno de los elementos más importantes, con el se llamará al objeto al incluirle en una sentencia.

- Posición del objeto (Position): en esta opción se determina la posición y el tamaño del control del objeto padre.

- Nombre del objeto (String): esta opción define el nombre que aparecerá en el control.

- Unidades (Units): unidades de medida en las que se interpretara el comando Position.

- Visible (Visible): indica si el control es visible en la ventana y puede tomar dos valores: on/off.

2.3.3.4. Tipos de uicontrol.

Existen ocho tipos de controles diferentes. La utilización de cada uno vendrá dada en función de sus características y aplicación:

1. Botones (Push Buttons y Toggle Buttons): al clicar sobre él con el ratón se producirá un evento que lanza una acción que deberá ser ejecutada por MatLab.

Figura 16. Push and Toggle Buttons.

2. Botones de selección (Check Boxes): los botones de selección permiten al usuario seleccionar entre las distintas opciones que aparezcan en la misma.

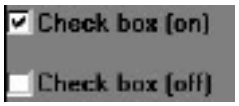


Figura 17. Check box.

3. Botones de opción (Radio Buttons): al igual que los botones selección, los botones de opción permiten al usuario seleccionar entre varias posibilidades. La diferencia reside en que los botones de opción excluyentes, es decir no puede haber más de uno activado, mientras que el control anterior permite tener dos o más cajas activadas.

Figura 18. Radio Buttons.

4. Barras de desplazamiento (Slider): permiten al usuario introducir un valor entre un rango de valores, que irán variando según se posicione la barra.

Figura 19. Slider.

5. Cajas de selección (Listbox): permite elegir una opción entre varias mostradas en una lista.

Figura 20. Listbox.

6. Cajas de texto (Static Text): son controles espaciales, ya que no permiten realizar ninguna operación con el ratón. Permite escribir un texto en la pantalla de modo informativo.

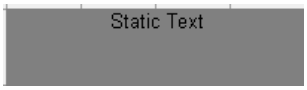


Figura 21. Static Text.

7. Cajas de texto editables (Edit Text): se emplean para introducir y modificar cadenas de caracteres, e incluso para mostrar datos al usuario. Su función editable dependerá de si su Enable se encuentra en estado On u Off.

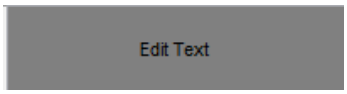


Figura 22. Edit text.

8. Marcos (Frames): no es un control propiamente dicho. Su función es la de englobar un serie de opciones (botones, cajas de texto, etc.) con el fin de mantener una estructura ordenada de controles.

Figura 23. Frames.

Estos serán los diferentes tipos de botones ejecutables que nos encontraremos a la hora de realizar una interfaz gráfica.

Por otra parte, cabe destacar la presencia de tablas y ejes que se podrán colocar en la interfaz con el fin de representar resultados obtenidos en dicha interfaz.

2.3.3.5. Creación de una interfaz gráfica

La creación de un GUIDE se podrá iniciar de dos maneras, escribiendo la palabra 'guide' en la ventana de comandos y pulsando intro o pulsando en la barra de herramientas: File-New-Gui. Al ejecutar una de las dos opciones nos aparecerá la siguiente ventana.

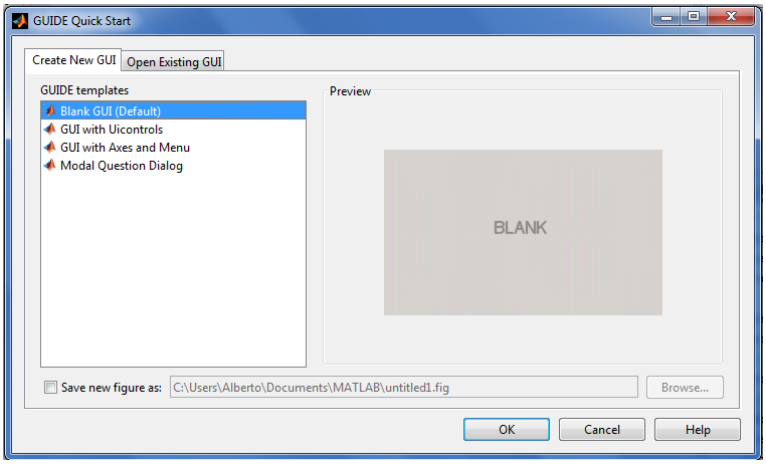


Figura 24. GUIDE quick start.

En ella, se nos proporciona la opción de crear una GUI desde el principio, totalmente vacía, varias opciones de modelos propuestos, o abrir una GUI ya existente que tengamos.

Si pulsamos la opción de abrir una GUI nueva nos aparecerá lo siguiente:

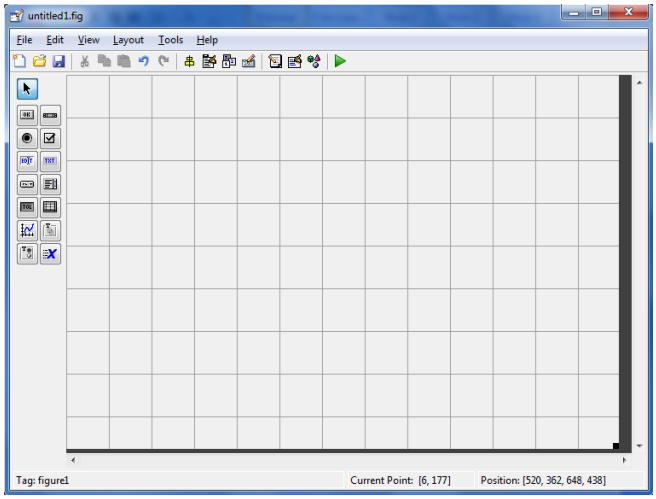


Figura 25. Generador de GUIs

A la izquierda de la imagen, observamos los uicontrol que se explicaron anteriormente, esa será la forma de seleccionarlo y escogerlos en función de las necesidades del GUI. Cada uno de los objetos que compongan el interfaz, podrán ser editados y modificados mediante el Property Inspector (Figura 26), donde se podrán observar todas las propiedades que tienen por defecto.

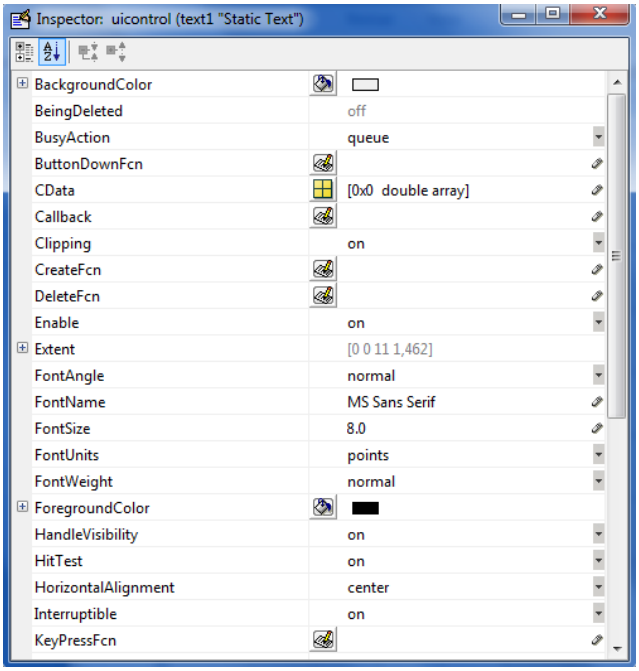


Figura 26. Property Inspector.

Una vez definidas las propiedades del GUI y sus componentes, se procederá a guardar dicha interfaz. Al guardarla se crearán dos archivos, un *.fig que contendrá la figura del GUI y por otro lado un archivo *.m que el código necesario para controlar los eventos lanzados por los diferentes controles de la interfaz.

2.4. Ed-Tridim

2.4.1. Introducción

Este software permite realizar el cálculo de vigas y pórticos hiperestáticos en dos o tres dimensiones, mediante el uso de métodos matriciales. El programa está organizado en cuatro módulos diferentes:

- Teoría: Este módulo está formado por un conjunto de lecciones teóricas que pueden ser consultadas por el usuario siempre que lo desee; las lecciones no pretenden ser exhaustivas sino tan sólo una ayuda para recordar conceptos básicos.

- Biblioteca: Permite introducir los datos que definen la estructura a analizar, es decir, la geometría, las propiedades de los materiales, las condiciones de contorno y las cargas que la solicitan; sirve igualmente tanto para crear nuevos problemas como para modificar los existentes.

- Ejemplos: En este módulo, el ordenador realiza paso a paso un problema sin que el usuario tenga que intervenir. El ordenador se convierte en guía, realizando todos los pasos, y deteniéndose en los aspectos más interesantes de cada etapa.
- Ejercicios: Este es el último módulo y a diferencia del anterior, se hace necesaria la participación activa del usuario. El ordenador asiste al usuario en el dibujo de las leyes de esfuerzos de la estructura.

2.4.2. Generación de un problema

Como se ha dicho con anterioridad, el módulo Biblioteca es el que permite generar una estructura de cualquier tipo. Al iniciar la sesión e introducirse en el módulo Biblioteca aparece el siguiente panel:

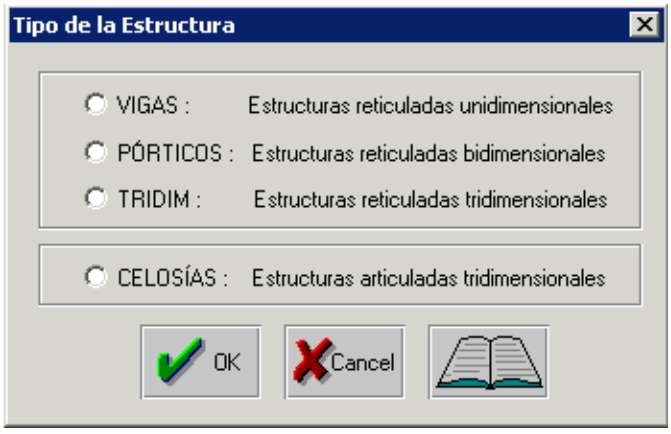


Figura 27. Seleccionador tipo de estructura

El usuario debe indicar el tipo de estructura que desea crear:

- Vigas: permite crear vigas planas unidimensionales;
- Pórticos: permite crear estructuras bidimensionales, ya sean articuladas o reticuladas;
- Tridim: permite realizar estructuras tridimensionales reticuladas;
- Celosía: en este último se pueden crear estructuras articuladas tridimensionales.

A continuación hay que dar las cotas espaciales máximas y mínimas de la estructura que se va a crear. Así mismo se puede crear una rejilla de cualquier paso para poder seleccionar con mayor comodidad los puntos de la estructura:

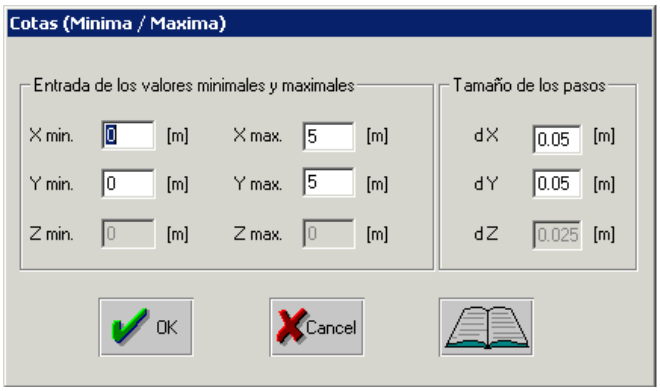


Figura 28. Selección de cotas.

Una vez definidas las características generales de la estructura, se procede a la definición concreta de cada uno de los componentes de ésta.

La barra de botones que aparece es la siguiente:

Figura 29. Barra de botones.

Los pasos que debemos seguir para elaborar cualquier tipo de estructuras son los siguientes:

1. Generación de los nodos: En primer lugar es necesario identifiar cuales son los nodos que componen la estructura; existen nodos de dos tipos: rígidos o articulados, por lo tanto además de la localización habrá que tener cuidado al seleccionar el tipo de nodo adecuado. La selección del tipo de nodo se hace teniendo presionado (o no) el botón de nodo articulado.

2. Generación de secciones y materiales: A continuación se tienen que identificar las propiedades del material y, o bien su geometría (en caso de ser una sección rectangular), o bien su momento de inercia.

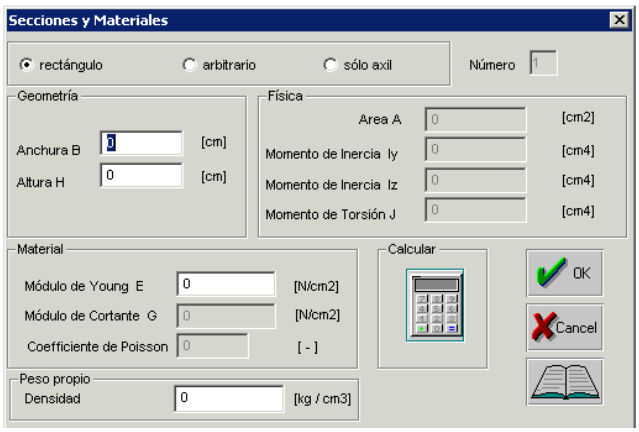


Figura 30. Creación de la sección.

Una vez que se ha definido un material, en la parte superior izquierda del cuadro de dibujo aparece un icono con la forma del tipo de sección elegida para ese material; en el caso de generar varios materiales habrá varios iconos, cada uno correspondiente a un material.

Pinchando con el ratón sobre dicho icono, se pueden ver las propiedades. Lógicamente el icono puede tener varias formas según se haya definido una sección arbitraria, una sección cuadrada, o una sección que va a trabajar sólo en axil.

3. Generación de barras: Una vez definidos el material o los materiales, se tienen que identificar las barras. Esta operación se realiza pinchando en los nodos que serán origen y final de la barra en cuestión.

Cuando se selecciona un nodo, éste cambia de color; de esta manera se puede saber si se está realizando bien la operación.

4. Generación de los apoyos: Este es el último apartado en la generación de la geometría de la estructura. En primer lugar se activa el icono de apoyos, después se selecciona el nodo en el que se va a colocar dicho apoyo y a continuación, aparecerá una ventana en la cual se tiene que indicar el tipo de apoyo que se quiere imponer.

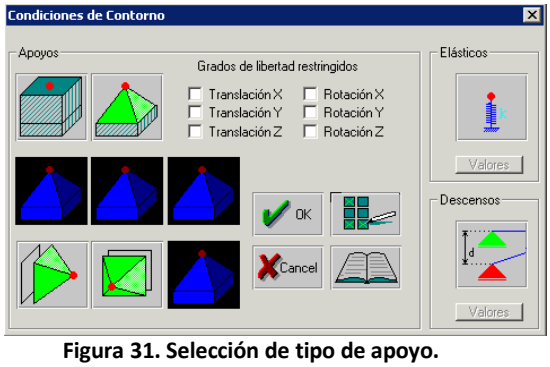


Figura 31. Selección de tipo de apoyo.

Una vez concluidos todos estos pasos, conviene salvar la estructura; esta operación se realiza como en cualquier programa que trabaja bajo Windows.

2.4.3. Visualización de resultados

Como se ha explicado anteriormente, los resultados se pueden visualizar utilizando el módulo de Ejercicios o el de Ejemplos. Ambos son muy similares, y la única diferencia estriba en que en el de Ejercicios se requiere de la participación del usuario, indicándole, de forma orientativa, la forma que tendrán los resultados, mientras que en el de Ejemplos no, el cual muestra todos los resultados obtenidos directamente.

En ambos casos el programa permite elegir como se va a resolver la estructura: con solución final o con solución paso a paso. En esta última modalidad, se van visualizando los cálculos que realiza el ordenador internamente.
Concretamente este software utiliza un método matricial, y por lo tanto nos muestra cómo se van formando las matrices de cada una de las partes de la estructura.

3. PROGRAMA INICIAL UCMM

En el siguiente punto, procederemos a la explicación del programa creado. En primer lugar mostraremos el programa tal y cómo lo elaboró mi compañero Alfonso Gago. Una vez establecidas las aportaciones y limitaciones del mismo, procederemos al desarrollo y explicación de las mejoras que han sido añadidas a dicho programa.

Para ayudarnos a hacer una descripción del mismo utilizaremos un organigrama, en el que se mostrarán los pasos generales que se deben seguir para llevar a cabo una buena ejecución del programa.

En términos generales, los pasos a seguir son:

1. Introducción de datos de la estructura.

2. Introducción del caso de carga.

3. Visualización de resultados.

3.1. Organigrama

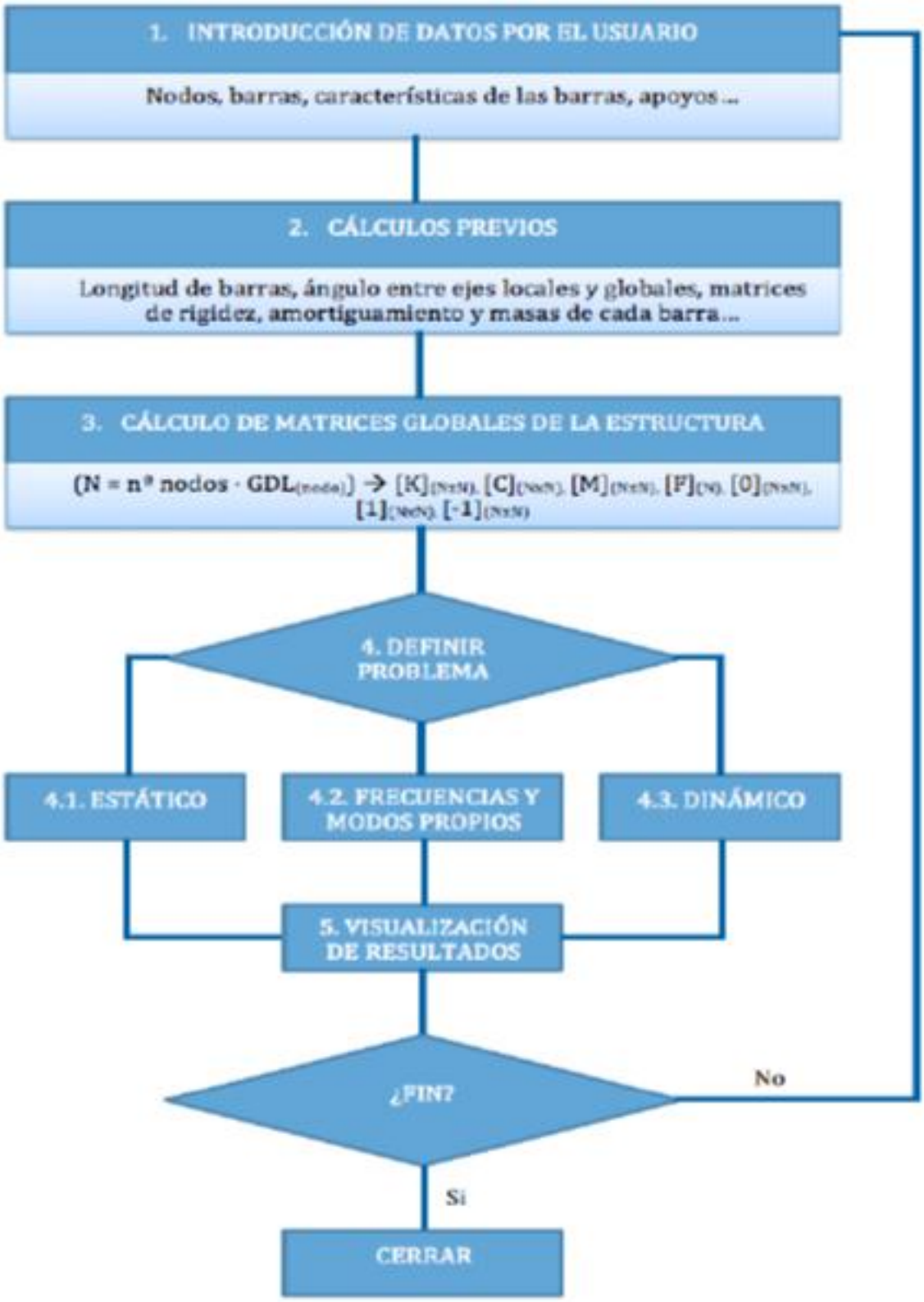


Figura 32. Organigrama UCMM.

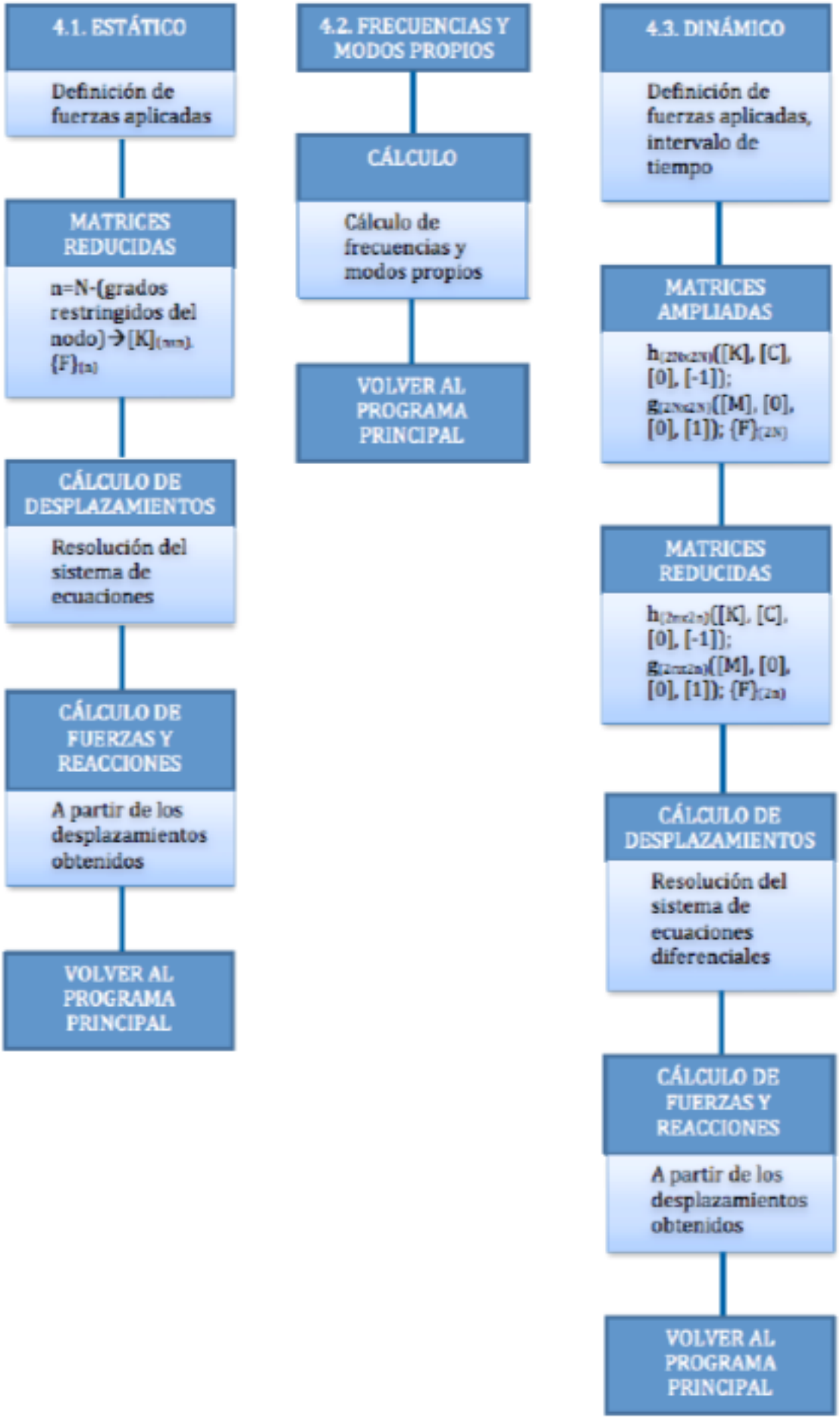


Figura 33. Desarrollo organigrama.

3.2 Procedimientos de cálculo

3.2.1.Introducción de datos

El primer paso necesario para la ejecución del programa será la introducción de datos por medio del usuario. Todos estos datos serán estrictamente necesarios para la obtención de desplazamientos y reacciones de la estructura. En esta parte cumplimos uno de los objetivos citados, en el que se programará de forma que no sea necesaria ninguna información redundante.

Los datos necesarios serán:

- Número de nodos de la estructura.
- Coordenadas de cada nodo.
- Número de barras de la estructura.
- Nodo menor y mayor, de cada una de las barras.
- Número de apoyos de la estructura.
- Tipos de apoyos que se colocarán.
- Número de muelles que tiene la estructura.
- Constante de rigidez de cada muelle y su posición en la estructura.
- Número de amortiguadores que tiene la estructura.
- Constante de amortiguamiento de cada amortiguador y su posición en la estructura.
- Tipos de secciones que tiene la estructura, indicando las características de cada una(densidad, módulo elástico, área y momento de inercia).

3.2.2. Cálculos previos

A partir de los datos introducidos por el usuario, el programa realizará unos cálculos generando datos de cada una de las barras, necesarios para la obtención de las matrices globales de la estructura:

- Longitud: a partir de las coordenadas del nodo menor y mayor de cada una de las barras y aplicando Pitágoras, el programa calculará las longitudes de las mismas.

- Ángulo entre ejes locales y globales: este ángulo se puede obtener conocidas las coordenadas de cada uno de los nodos de cada barra de la estructura, mediante relaciones trigonométricas.

- Matrices de rigidez: a partir de las características que tenga la sección de cada barra, su longitud, el ángulo entre ejes globales y locales, y, la constante de rigidez de los muelles que tenga la estructura. Se realizará un cálculo de la matriz de rigidez de cada una de las barras en ejes globales.

- Matrices de amortiguamiento: a partir de la constante de amortiguamiento de los amortiguadores que tenga la estructura, se calculará las matrices de amortiguamiento de las mismas.

- Matrices de masas: cada barra repartirá su peso equitativamente entre sus dos nodos, de forma que, se generará una matriz por nodo con la masa que soportará cada uno.

Todas las matrices citadas anteriormente serán de tamaño 2x2 para el caso de estructuras articuladas y de 3x3 en estructuras reticuladas.

3.2.3. Cálculo de matrices globales

Conocidas las matrices de rigidez y amortiguamiento de cada barra, y matriz de masas de cada nodo de la estructura, el programa calculará con estos datos, las matrices globales de la estructura:

- Matriz de rigidez global de la estructura: conocidas las matrices de rigidez de cada barra y los nodos que tiene cada una, el programa ubicará cada una de las submatrices de cada barra, en la posición que le corresponda en la matriz de rigidez global. Ésta será una matriz cuadrada, con tantas celdillas como número de nodos tenga la estructura.

  Cada submatriz será de 3x3 en el caso de estructuras reticuladas, y 2x2 en el caso de estructuras articuladas, es decir, que la matriz de rigidez global de la estructura, tendrá unas dimensiones NxN, teniendo en cuenta que:

  N = (nº de nodos de la estructura) * (grados de libertad del nodo)

- Matriz de amortiguamiento global de la estructura: los conceptos y procedimiento de cálculo son exactamente los mismos que para la matriz de rigidez global de la estructura, con una diferencia, y es que, en este caso, la matriz de amortiguamiento global de la estructura, se calculará a partir de las matrices de amortiguamiento de cada una de las barras.

- Matriz de masas global de la estructura: consiste en una matriz en la que cada posición de la diagonal principal esta ocupada por cada una de las matrices de masas de cada nodo, siendo el resto de los valores ceros.

Por otro lado, también se generarán otras tres matrices numéricas:

- Matriz de ceros: todos los valores de la matriz serán cero.
- Matriz de unos: todos los valores de la matriz serán cero menos los de la diagonal principal, cuyos valores serán uno.
- Matriz de menos unos: todos los valores de la matriz serán cero menos los de la diagonal principal, cuyos valores serán menos uno.

Tendrán la misma dimensión que las matrices de rigidez, amortiguamiento y masas. Solo serán necesarias en la resolución de estructuras reticuladas, para conformar las matrices ampliadas.

Como se ha explicado previamente en antecedentes, a la hora de resolver una estructura (ya sea articulada o reticulada) sometida a cargas estáticas, se planteará la ecuación matricial a partir de la cual se obtendrá un sistema de ecuaciones que habrá que resolver.

$$\{F\}_N = [K]_{(NxN)} * \{X\}_N$$

(39)

Siendo:

$\{F\}$ El vector de fuerzas

$[K]$ La matriz de rigidez global de la estructura

$\{X\}$ El vector desplazamientos

3.2.4. Cálculo matrices ampliadas

Cuando se efectúa un análisis dinámico, las acciones sobre la estructura son función del tiempo, lo que hace movilizar unas fuerzas de inercia unidas a las masas y a la aceleración, planteándose la siguiente ecuación en forma matricial:

$$\{F(t)\}_N = ([K]_{(NxN)} * \{X\}_N) + [M]_{(NxN)} * \{\ddot{X}\}_N)$$

(40)

Siendo:

$\{F(t)\}$ El vector de fuerzas aplicadas

$[M]$ La matriz de masas global de la estructura

$\{\ddot{X}\}$ El vector de aceleración

De forma análoga si tuviéramos un amortiguador en la estructura, la ecuación resultante tendría la siguiente forma:

$$\{F(t)\}_N = ([K]_{(NxN)} * \{X\}_N) + ([M]_{(NxN)} * \{\ddot{X}\}_N) + ([C]_{(NxN)} * \{\dot{X}\}_N)$$

(41)

Siendo:

$[C]$ La matriz de amortiguamiento global de la estructura

$\{\dot{X}\}$ El vector de velocidades

Observamos que aparece una ecuación de segundo orden, por lo que nos vemos en la necesidad de convertirla mediante un cambio de variable. Si se plantea de forma matricial este sistema de ecuaciones y cambios de variables, se obtendrán unas matrices ampliadas que estarán formadas por las matrices globales de la estructura, y las matrices de ceros, unos y menos unos, que permitirán plantear los cambios de variables. Quedando de la siguiente forma:

$$\underbrace{\begin{bmatrix} \{0\}_{(N)} \\ \{F(t)\}_{(N)} \end{bmatrix}}_{\{F(t)\}_{(2N)}} = \underbrace{\begin{bmatrix} [1]_{(NxN)} & [0]_{(NxN)} \\ [0]_{(NxN)} & [M]_{(NxN)} \end{bmatrix}}_{[g]_{(2Nx2N)}} \cdot \underbrace{\begin{bmatrix} \{\dot{X}_1\}_{(N)} \\ \{\dot{X}_2\}_{(N)} \end{bmatrix}}_{\{\dot{X}\}_{(2N)}} + \underbrace{\begin{bmatrix} [0]_{(NxN)} & [-1]_{(NxN)} \\ [K]_{(NxN)} & [C]_{(NxN)} \end{bmatrix}}_{[h]_{(2Nx2N)}} \cdot \underbrace{\begin{bmatrix} \{X_1\}_{(N)} \\ \{X_2\}_{(N)} \end{bmatrix}}_{\{X\}_{(2N)}}$$

Por tanto, el programa se encargará de generar, a partir de las matrices globales de la estructura calculadas previamente, las matrices ampliadas "$[g]_{(NxN)}$" y "$[h]_{(NxN)}$".

Con este planteamiento genérico se podrá resolver estructuras de "n" grados de libertad.

3.2.5. Definición de Cargas aplicadas

Llegados a este punto, lo único que quedaría por definir sería el tipo de cargas que se tendrían que aplicar a la estructura. Trabajaremos con los siguientes tipos:

- Cargas estáticas.
- Cargas dinámicas.

3.2.5.1. Cargas estáticas

Será necesario que el programa genere un vector de fuerzas $\{F\}_N$, tal que, el primer valor del vector será un vector que se corresponderá con el nodo uno, el segundo valor con el nodo dos y así sucesivamente.

El vector de cada nodo estará formado por tres o dos valores dependiendo del tipo de estructura (dos en articuladas y tres en reticuladas). El primer valor del vector de un nodo indicará la magnitud de la fuerza en el eje "x", el segundo será la magnitud de la fuerza en el eje "y" y en el caso de estructuras reticuladas, el tercero será la magnitud del momento aplicado en el eje "z".

Para ello el usuario deberá indicar en qué nodos y con qué magnitud, dirección y sentido se va a aplicar la fuerza en cada uno de ellos.

El programa únicamente permitirá introducir estas fuerzas aplicadas en los nodos, y una vez indicadas dichas fuerzas se encargará de ubicar esos valores en el vector de ceros $\{F\}_N$.

De forma que el vector de fuerzas generado para una barra de una estructura reticulada aplicándose una fuerza en el nodo uno, en dirección "y" y en sentido contrario a los ejes de coordenadas será el siguiente:

$$\{F\}_N^t = (0 \; -F \;\; 0 \;\; 0 \;\; 0 \;\; 0)$$

(42)

3.2.5.2. Cargas dinámicas

En el caso de someter a la estructura a cargas dinámicas, además de indicar los mismos datos que en cargas estáticas, se tendrá que introducir la frecuencia angular (ω), el desfase (φ) y, teniendo en cuenta que la carga dinámica depende del tiempo, será necesario introducir el intervalo de tiempo en el que se quiere estudiar la respuesta de la estructura sometida a dichas cargas.

De forma que para el ejemplo planteado en el apartado anterior, se obtendría el siguiente vector de fuerzas:

$$\{F(t)\}_N^t = (0 \; -F * sin\{(\omega * t) + \varphi\} \;\; 0 \;\; 0 \;\; 0 \;\; 0)$$

(43)

3.2.6. Obtención de Matrices reducidas

Como se comentó anteriormente en los Antecedentes, para la obtención de desplazamientos, fuerzas y reacciones, habrá que plantear en primer lugar, unas ecuaciones de las que se obtendrán los desplazamientos de la estructura, y posteriormente, conocidos los desplazamientos, se obtendrán las fuerzas y reacciones de la estructura.

Se partirá de una ecuación matricial en la que se contará con todos los grados de libertad de los nodos de la estructura (libres y restringidos). Para la obtención de desplazamientos, será necesario eliminar todas las filas y columnas en las que coincida que el grado de un nodo este restringido, es decir, que su desplazamiento sea nulo. Por lo tanto, habrá que reducir en el caso de estructuras sometidas a cargas estáticas, la matriz de rigidez global de la estructura $[K]_{(NxN)}$ y su vector de fuerzas $\{F\}_N$ o, en el caso de estructuras sometidas a cargas dinámicas, las matrices ampliadas de la estructura $[g]_{(NxN)}$ y $[h]_{(NxN)}$ y su vector de fuerzas ($\{F(t)\}_N^t$).

De forma que la dimensión de las matrices y vectores reducidos será:

n = N – (GDL restringidos de cada nodo)

El programa será capaz, a partir de los datos introducidos por el usuario en los que indica el número de apoyos y sus restricciones, calcular las matrices reducidas de la matriz de rigidez global $[k]_{(nxn)}$, su vector de fuerzas $\{f\}_n$, las matrices ampliadas $[G]_{(nxn)}$ y $[H]_{(nxn)}$ y el vector de fuerzas $\{f(t)\}_n^t$ de la estructura.

3.3. Resultados obtenidos

Se podrán mostrar los resultados numéricos de los valores y modos propios, desplazamientos, fuerzas y reacciones, de cualquier tipo de estructura sometida a cualquier tipo de carga.

Además, para estructuras sometidas a cargas dinámicas, se podrá mostrar en gráficas, los resultados de los desplazamientos, fuerzas y reacciones de la estructura frente al tiempo. Pudiéndose comparar desplazamientos y fuerzas de distintos nodos de la estructura, e incluso, comparar desplazamientos de nodos de estructuras diferentes.

3.3.1. Obtención de valores y modos propios

Cuando un cuerpo capaz de vibrar es sometido a la acción de una fuerza periódica, cuyo periodo de vibración coincide con el periodo de vibración característico de dicho cuerpo, se produce la resonancia.

Por lo que, una fuerza relativamente pequeña aplicada en forma repetida, hace que la amplitud de un sistema oscilante se haga muy grande.

De forma algebraica, hallaremos los valores propios que definen la frecuencia y los periodos naturales de la estructura mediante el siguiente determinante:

$$\|[K][M]^{-1} - \omega\| = 0$$

(44)

Una vez obtenidos éstos, la siguiente formula nos proporcionará la forma modal correspondiente a cada valor propio, que será un modo propio para cada valor, que representara las amplitudes de la deformada de cada modo de vibración.

$$([K] - \omega^2 * [M]) * x = 0$$

(45)

Hay tantos modos de vibración como grados de libertad de la estructura considerada.

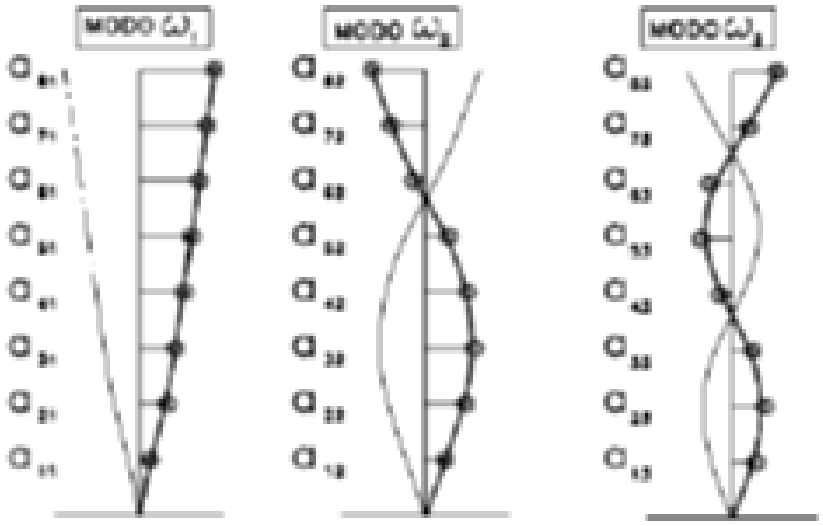


Figura 34.Primeros modos de vibración.

3.3.2. Obtención de desplazamientos

En el caso de estructuras sometidas a cargas estáticas, el programa tendrá planteada una ecuación en la que se despeja el vector $\{x\}_{(n)}$, y se obtienen los resultados de los desplazamientos:

$$\{x\}_{(n)} = [k]^{-1}_{(nxn)} * \{f\}_n$$

(46)

Por el contrario, si nos encontramos ante un problema cargado con fuerzas dinámicas, tendremos que operar de la siguiente forma:

$$\{\dot{x}\}_{(n)} = [G]^{-1}_{(nxn)} * (\{f(t)\}_{(n)} - [H]_{(nxn)} * \{x\}_{(n)})$$

(47)

3.3.3. Obtención de reacciones

Para el cálculo de las reacciones y fuerzas de la estructura, el programa creará un vector desplazamientos $\{x\}_{(n)}$, a partir de los desplazamientos obtenidos en el paso previo, y en el que se encontrarán tanto los desplazamientos de los nodos no restringidos, como el de los nodos restringidos (apoyos), cuyo valor seria cero. De forma que, se podrá obtener el vector de fuerzas $\{F\}_{(n)}$, que contendrá el valor de las fuerzas y reacciones generadas en cada nodo de la estructura.

Esto será posible mediante las siguientes ecuaciones en función de si trabajamos con cargas estáticas o dinámicas respectivamente:

$$\{F\}_N = [K]_{(NxN)} * \{X\}_N$$

(48)

$$\{F(t)\}_N = ([K]_{(NxN)} * \{X\}_N) + ([M]_{(NxN)} * \{\ddot{X}\}_N) + ([C]_{(NxN)} * \{\dot{X}\}_N)$$

(49)

3.4. Limitaciones UCMM

Analizando los tres módulos generales de los que consta el formato del programa UCMM, se puede comprobar que los más limitados son la inserción de fuerzas y la visualización de los resultados.

En el caso de la introducción de cargas, estas sólo podrán ser nodales, viéndonos obligados a hacer una descomposición de cualquier tipo de fuerza no nodal que nos encontremos en la estructura que deseemos resolver.

Por otro lado, la visualización de resultados que ofrece es muy escueta y poco intuitiva. En ella únicamente podremos observar el vector de fuerzas y el vector de desplazamientos. De cara al análisis de una estructura, existen muchos resultados de interés que se pueden calcular, como el diagrama de momentos flectores, la flecha o l distribución de tensiones en cualquier viga.

4. PROGRAMA MEJORADO UCMM v2.0

4.1. Organigrama

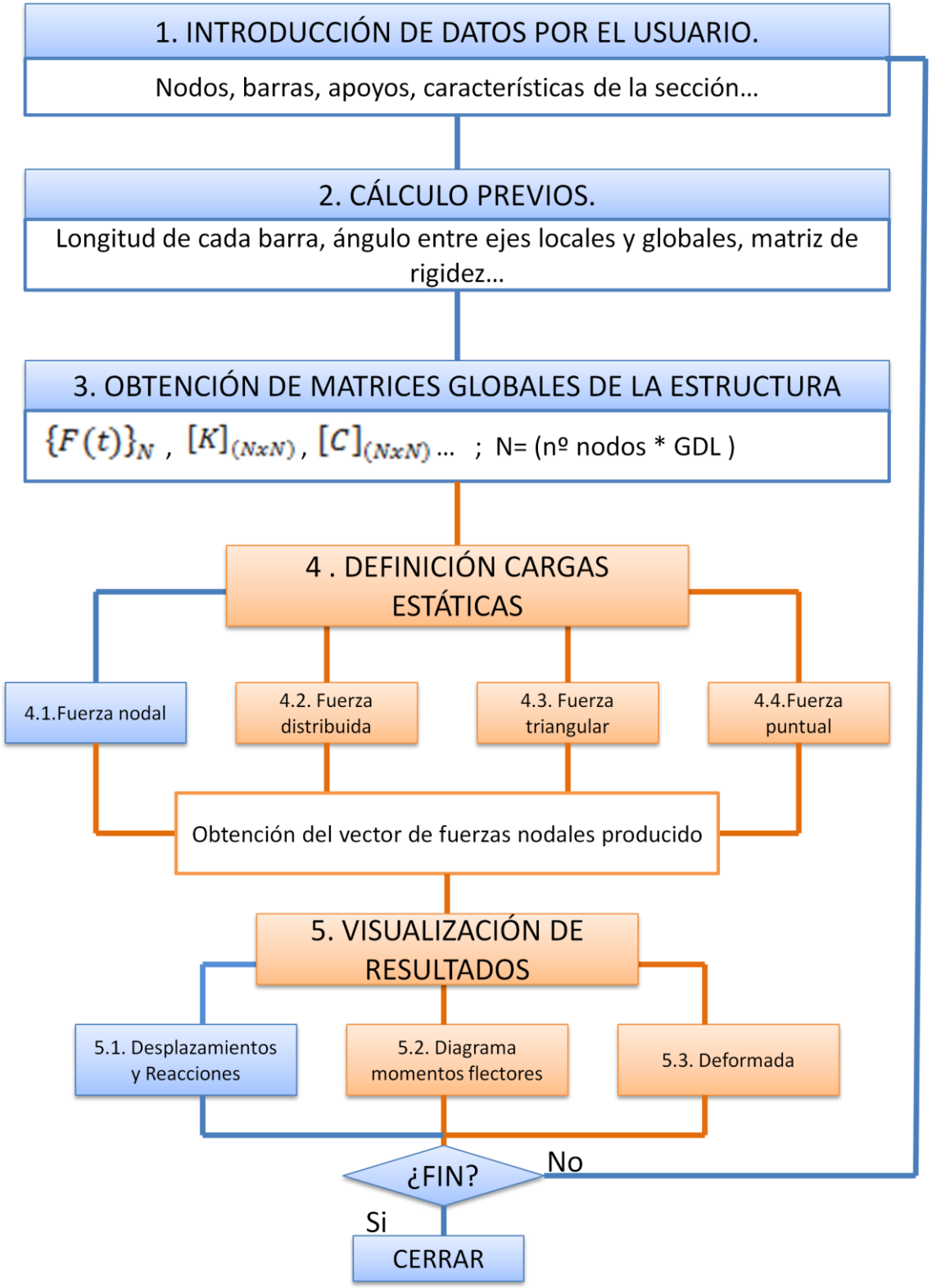


Figura 35. Organigrama UCMMv2.0.

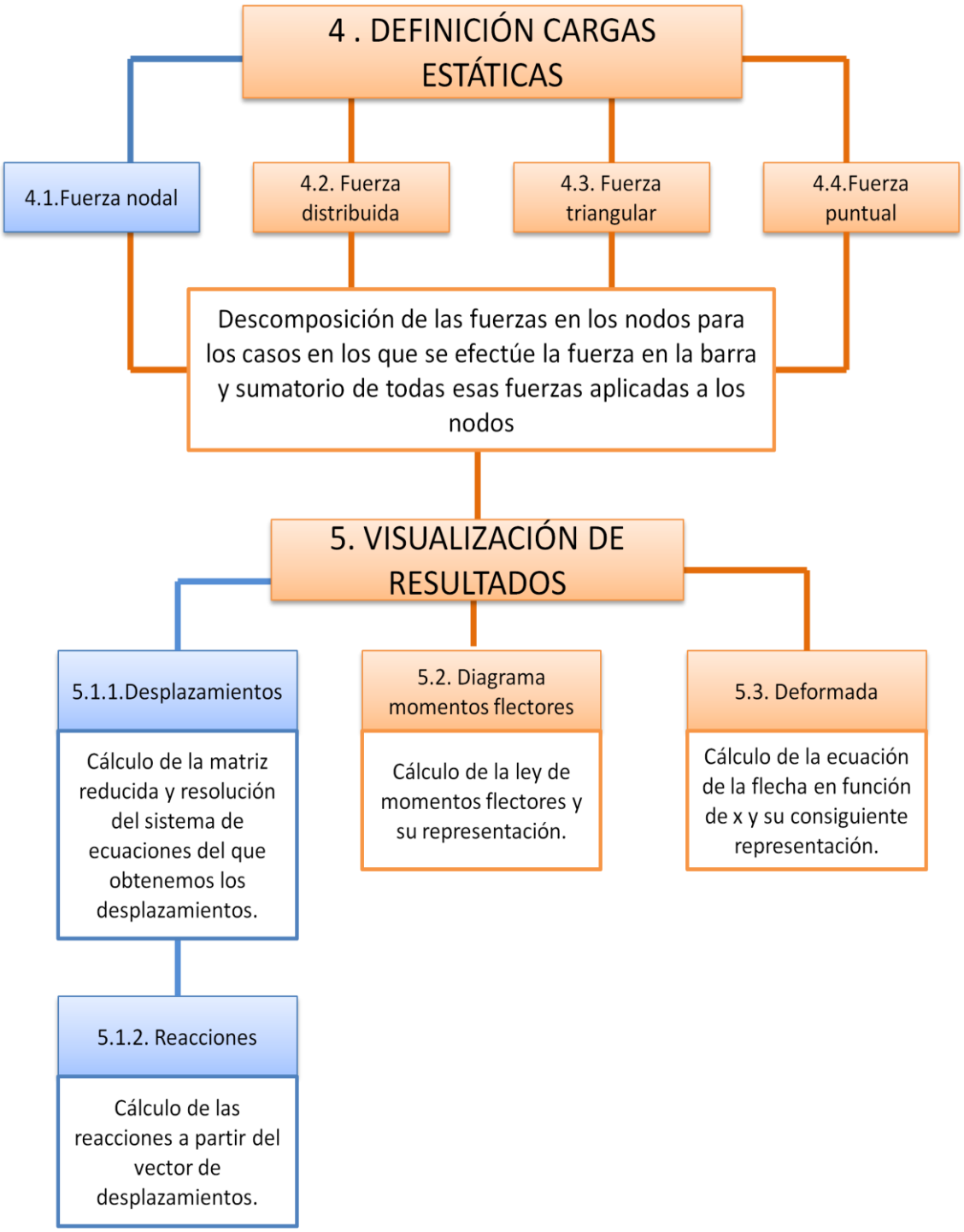


Figura 36. Desarrollo organigrama UCMMv2.0.

Las partes azules de los organigramas representan las partes del programa correspondiente al UCMM, mientras que las mejoras, es decir, el denominado UCMMv2.0 son las partes de color naranja.

4.2. Resumen

En este apartado se tratarán de explicar las mejoras que se le realizaron al UCMM. Para ello, intentaremos observar los puntos débiles del mismo e intentaremos fortalecerlos para así conseguir un programa más completo y versátil.

El programa trabajará de la misma manera y con el mismo formato que hasta ahora, es decir, evitando la inserción redundante de datos y con un manejo fácil e intuitivo.

Los archivos programados que abrirán las posibilidades, en ningún caso interferirán con la manera de resolución establecida hasta ahora, siendo solo adiciones a lo que ya produce el código.

Todas las mejoras llevadas a cabo serán iguales tanto para estructura articuladas como reticuladas.

Los dos apartados en los que se centrará el desarrollo del UCMMv2.0 serán la ampliación de tipos de cargas y la visualización de resultados.

La interfaz se habilitará de modo que el usuario tenga la posibilidad de introducir cargas de los siguientes tipos:

- Uniformemente distribuidas.
- Triangularmente distribuidas.
- Puntuales en cualquier punto de la barra y perpendicular a esta.

Por otro lado, la ampliación en la visualización de resultados se llevará a cabo en:

- Una representación de la ley de momentos flectores.
- Una representación de la deformada de la estructura.

4.3. Descripción de las mejoras

4.3.1. Mejoras en la parte de inserción de datos

Como ya se ha comentado anteriormente, el programa hasta ahora, solo permite introducir fuerzas nodales, es decir, aplicadas en los nodos, ya establecidos, de la estructura.

Esta forma de trabajar, requiere hacer una descomposición previa de dichas fuerzas para así poder resolver la estructura deseada.

De cara a establecer una posible solución, se consideró de gran utilidad, habilitar el programa de forma que tuviera la opción de aplicar a la estructura fuerzas que se situaran en las barras.

Se consideró de prioridad los tres tipos de fuerzas que han sido habilitados debido a que en general son los más utilizados a la hora de la resolución de estructuras.

Estos tipos son:

- Cargas uniformemente distribuidas a lo largo de toda la barra.
- Cargas triangularmente distribuidas a lo largo de toda la barra.
- Carga puntual colocada en cualquier posición de la barra y perpendicular a esta.

Analizaremos la forma en que se atacó el problema y la forma de interpretarlas con el programa tanto para estructuras articuladas como reticuladas.

4.3.1.1. Estructuras articuladas:

Una estructura se dice articulada o triangulada cuando está formada por barras conectadas entre si mediante articulaciones perfectas (rótulas).

Mostraremos una barra articulada que va del nodo A al nodo B y analizaremos su comportamiento:

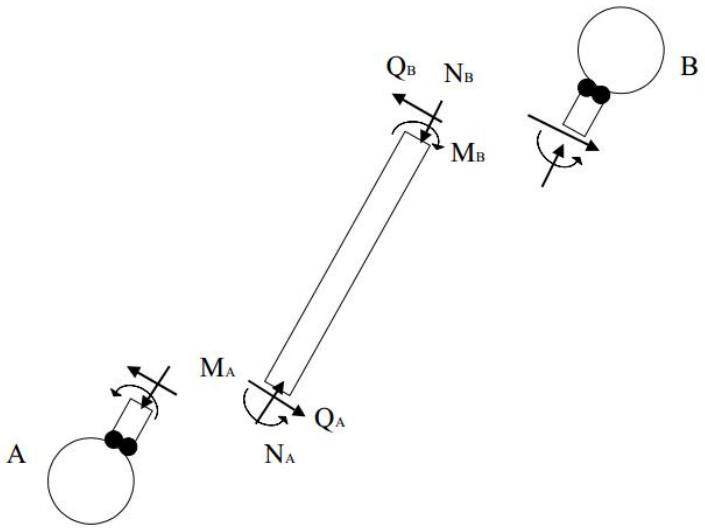


Figura 37. Descomposición barra articulada.

Analizando el equilibrio de la rótula sometida a los esfuerzos que la barra le transmite se obtienen las siguientes conclusiones:

- M_A (=M_B) = 0 porque la rótula no absorbe momentos
- Q_A (=Q_B) = 0 porque de no ser así, sobre la barra existiría un par que habría de ser equilibrado con un momento
- N_A = $N_B \neq 0$

Obsérvese en la figura que si el axil de la barra es de compresión, la acción de la barra contra el nudo lleva la dirección de barra a nudo. De forma contraria pasaría si el axil fuese a tracción.

Debido a este fenómeno, cuando en una estructura articulada, nos encontramos con una fuerza fuera de los nodos, es decir, en cualquier barra, como entre estas sólo se transmite el axil operaremos de la siguiente manera:

1. Aislamos la barra cargada y colocamos en los nodos, dos apoyos en los que sabemos que el momento será nulo.
2. Hallamos las 'reacciones' que se generarán en mencionados apoyos.

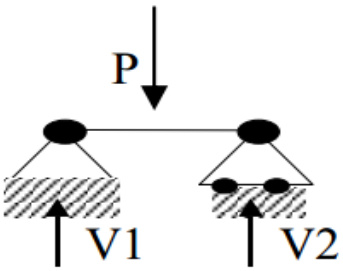


Figura 38. Reacciones generadas.

3. Colocaremos las reacciones halladas (V1 y V2) pero con el sentido cambiado en la estructura original.

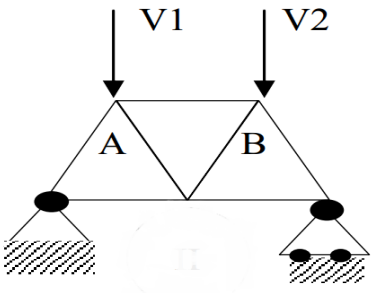


Figura 39. Fuerzas aplicadas.

4. Llegado a este punto se opera tal y como se explica en el apartado de Cálculo matricial en Antecedentes.

Con estas premisas, analizaremos el procedimiento que llevamos a cabo para crear el algoritmo que utilizaremos más adelante en la programación en función de los tres tipos de carga que analizaremos:

En todos los casos, al analizar una estructura isostática, plantearemos las ecuaciones de equilibrio y hallaremos el valor de las reacciones que se generan en función del valor de la carga aplicada.

a) Carga uniformemente distribuida:

Analizando tal y como se explico anteriormente, hallaremos las reacciones para los nudos inicial (A) y final (B).

$$R_A = R_B = \frac{P * L}{2} \quad (50)$$

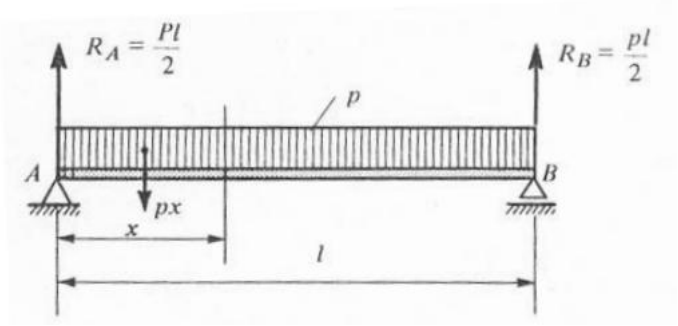


Figura 40. Carga distribuida uniforme

Siendo:

P: El valor de la carga aplicada.

L: La longitud total de la barra.

b) Carga triangularmente distribuida:
Procediendo de forma análoga al caso anterior obtenemos:

$$R_A = \frac{P_{max}}{3}$$

$$R_B = \frac{2 * P_{max}}{3} \quad (51)$$

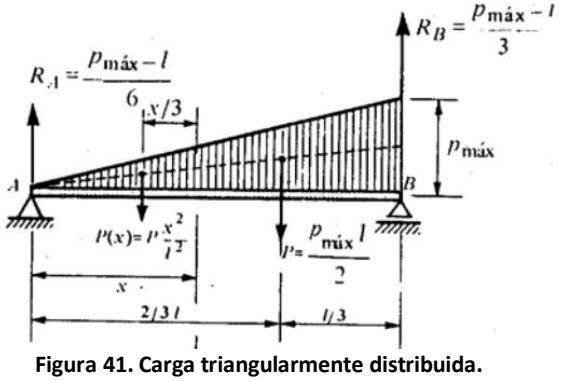


Figura 41. Carga triangularmente distribuida.

Siendo:

P_{max}: El valor máximo de la carga aplicada, es decir, la altura del triangulo, que estará siempre colocado en el nodo final (B)

c) Carga puntual situada a una distancia 'a' de la barra:

En este caso, primeramente, especificar que la distancia a es la distancia que hay desde el nodo inicial (A) hasta el punto de aplicación de la barra. Siempre consideraremos esta carga aplicada en dirección perpendicular a la barra.

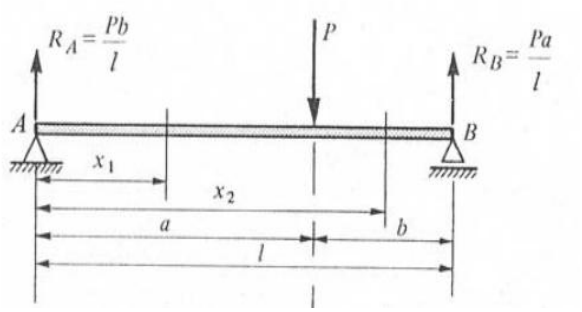


Figura 42. Carga puntual.

$$R_A = \frac{P * b}{L}$$

$$R_B = \frac{P * a}{L}$$

(52)

Siendo:

P: El valor de la carga aplicada

L: La longitud de la barra

b: La distancia desde el nodo final (B) hasta el punto de aplicación de la carga.

En los tres casos, cuando el programa se encuentre con una carga de estas, hallará dichas reacciones indicadas, les cambiará el sentido y se las añadirá al vector de fuerzas en los nodos $\{F\}_N$ para que continúe la resolución de la estructura tal y como estaba establecido.

4.3.1.2. Estructuras reticuladas

De forma análoga que en estructuras articuladas, descompondremos las fuerzas aplicadas en las barras en fuerzas nodales.

A diferencia de las estructuras articuladas, los nudos con los que nos encontramos en reticuladas, si que transmiten todo tipo de esfuerzo, por lo que tendremos cortante y momento flector en dichos nodos.

Para calcular el valor de esas fuerzas nodales, trataremos los mismos tres casos desarrollados anteriormente pero cambiando los apoyos por empotramientos.

a) Carga uniformemente distribuida:

Analizando tal y como se explico anteriormente, hallaremos las reacciones para los nudos inicial (A) y final (B).

$$R_A = R_B = \frac{P * L}{2}$$

$$M_A = M_B = \frac{P * L^2}{12}$$

(53) **Figura 43. Carga distribuida uniforme viga biempotrada.**

Siendo:

P: El valor de la carga aplicada.

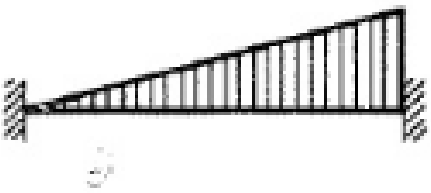
L: La longitud total de la barra.

b) Carga triangularmente distribuida:

Procediendo de forma análoga al caso anterior obtenemos:

$$R_A = \frac{3 * P_{max}}{20} \qquad M_A = \frac{P_{max} * L^2}{30}$$

$$R_B = \frac{7 * P_{max}}{20} \qquad M_B = \frac{P_{max} * L^2}{20}$$



(54) **Figura 44. Carga triangularmente distribuida viga biempotrada.**

Siendo:

P_{max}: El valor máximo de la carga aplicada, es decir, la altura del triangulo, que estará siempre colocado en el nodo final (B)

c) Carga puntual situada a una distancia 'a' de la barra:

De igual forma que ocurría en estructuras articuladas, la distancia a es la distancia que hay desde el nodo inicial (A) hasta el punto de aplicación de la barra. Siempre consideraremos esta carga aplicada en dirección perpendicular a la barra.

$$R_A = \frac{P * b^2}{L^3} (2a + L)$$

$$R_B = \frac{P * a^2}{L^3} (2b + L)$$

Figura 45. Carga puntual viga biempotrada.

$$M_A = \frac{Pab^2}{L^2}$$

$$M_B = \frac{Pba^2}{L^2}$$

(55)

Siendo:

P: El valor de la carga aplicada

L: La longitud de la barra

b: La distancia desde el nodo final (B) hasta el punto de aplicación de la carga.

En los tres casos, de la misma forma que se procede en articuladas, cuando el programa se encuentre con una carga de estas, hallará dichas reacciones indicadas, les cambiará el sentido y se las añadirá al vector de fuerzas en los nodos $\{F\}_N$ para que continúe la resolución de la estructura tal y como estaba establecido.

4.3.2. Mejoras en la obtención de resultados

El segundo aspecto que se consideró necesario mejorar, fue la visualización de resultados. Hasta ahora, el programa únicamente nos permitía obtener los desplazamientos producidos en los nodos con movimientos no restringidos y las reacciones producidas en los apoyos de la estructura.

La mejora introducida nos permitirá visualizar dos aspectos considerados de gran importancia en la resolución de estructuras. Estos son:

- Representación de los momentos flectores en la estructura.
- Representación de la deformada y el desplazamiento de los nodos.

En los siguientes apartados explicaremos el desarrollo seguido para la obtención de dichas mejoras para ambos tipos de estructuras.

4.3.2.1. Estructuras articuladas

Para el análisis de estas estructuras nos basaremos en dos conceptos que fueron tratados con anterioridad en este documento.

Por un lado el método de la doble integración, que utilizaremos obtener tanto el momento flector como la ecuación de la flecha. Por otro lado la característica principal de las estructuras de este tipo: sus uniones mediante rótulas.

Analizaremos la obtención de la flecha y el momento para cada uno de los tipos de cargas programados:

a) Carga uniformemente distribuida:

El momento flector en vigas cargadas con este tipo de carga en función de x responderá a la siguiente forma:

$$M_x = R_A \cdot x - P \cdot x \cdot \frac{x}{2}$$

$$M_x = \frac{P * L}{2} \cdot x - \frac{P x^2}{2} \quad (56)$$

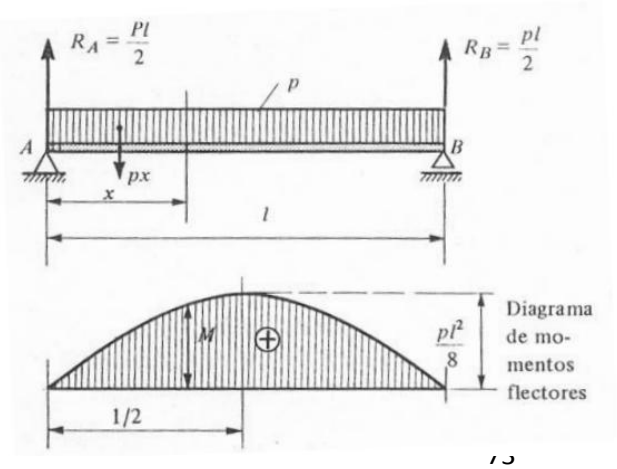


Figura 46. Momento flector viga biapoyada carga distribuida.

Derivando esta función e igualándola a 0, obtenemos los puntos donde su pendiente es igual a 0, es decir, los máximos.

$$\frac{\partial M}{\partial x} = 0 \quad \rightarrow \quad x_{max} = \frac{L}{2}$$

(57)

Por lo que el momento en ese punto será donde obtengamos su valor máximo.

$$M_{max} = M\left({}^{L}/_{2}\right) = \frac{P * L^2}{8}$$

(58)

Como ya vimos con anterioridad, al integrar la ecuación general del momento flector, obtendremos la ecuación de la pendiente:

$$EI\frac{dy}{dx} = \int \frac{P * L}{2} \cdot x - \frac{Px^2}{2} dx = \frac{P * L}{4} \cdot x^2 - \frac{Px^3}{6} + C_1$$

(59)

Sabemos que la pendiente de la viga será nula en el punto en el que el momento flector es máximo, L/2:

$$\frac{P*L}{4} \cdot \frac{L}{4}^2 - \frac{P\left({}^{L}/_{2}\right)^3}{6} + C_1 = 0 \qquad \rightarrow \qquad C_1 = -\frac{PL^3}{24}$$

(60)

Integrando ahora la ecuación de la elástica obtendremos la ecuación general de la flecha:

$$EI \cdot y = \int \frac{P*L}{4} \cdot x^2 - \frac{Px^3}{6} - \frac{PL^3}{24}\, dx$$

$$y = \frac{PLx^3}{12} - \frac{Px^4}{24} - \frac{PxL^3}{24} + C_2$$

(61)

En este caso, sabemos que la flecha será nula en el apoyo, es decir cuando x=0, por lo que deducimos que $C_2 = 0$, obteniendo la ecuación que nos proporcionará la flecha en vigas cargadas con una carga uniformemente distribuida.

$$EIy = \frac{PLx^3}{12} - \frac{Px^4}{24} - \frac{PxL^3}{24}$$

(62)

b) Carga triangularmente distribuida:

Haciendo un análisis idéntico al caso estudiado anteriormente, obtenemos:

$$M_x = \frac{Px}{3} - \frac{Px^3}{3L^2} \quad (63)$$

$$x_{max} = \frac{L}{\sqrt{3}}$$

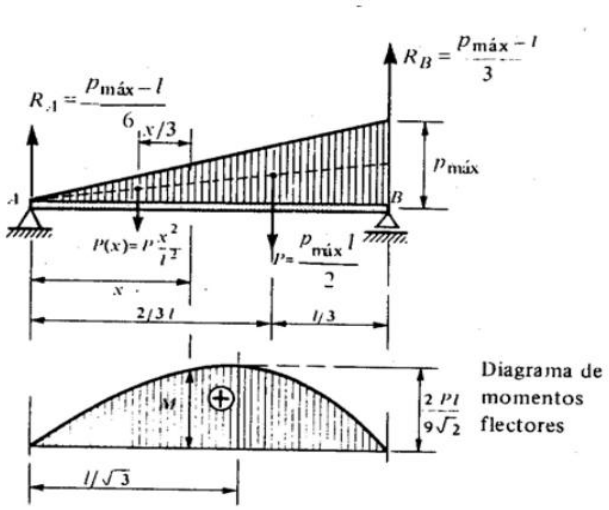


Figura 47.Momento flector viga biapoyada carga triangular.

$$M_{max} = M\left({}^{L}/_{\sqrt{3}}\right) = \frac{2PL}{9\sqrt{2}} \quad (64)$$

Siendo para todas las ecuaciones:

L: La longitud de la barra

$$P = \frac{P_{max} * L}{2}$$

Del mismo modo, obtendremos la ecuación de la flecha, que será:

$$y = \frac{PxL^3}{360EI}\left(7 - \frac{x^2}{L^2} + \frac{3x^4}{L^4}\right) \quad (65)$$

c) Carga puntual situada a una distancia 'a' del nodo inicial.

En este caso, la distancia 'a' será la existente a lo largo de la viga desde el nodo inicial hasta el punto de aplicación de la carga. Por el contrario, la distancia 'b' será lo que reste a la longitud total de la viga.

Con estas premisas, desarrollamos este caso de la misma forma que se hizo en el caso de carga distribuida uniforma mostrado anteriormente.

El momento flector de la parte anterior a la viga cambiará en cuanto a la posterior.

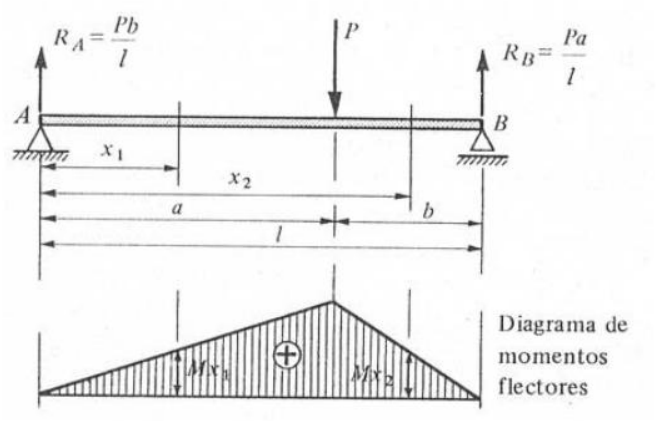


Figura 48.Momento flector viga biapoyada carga puntual.

$$M_x = R_A x = \frac{P * b}{L} x \qquad para \qquad 0 \leq x \leq a$$

$$M_x = R_A x - P(x - a) = \frac{P * a}{L}(L - x) \qquad para \qquad a \leq x \leq L$$

(66)

En este caso, tendremos el máximo momento flector coincidente con el punto de aplicación de la carga, es decir, cuando x=a.

$$M_{max} = M(a) = \frac{P * a * b}{L}$$

(67)

Siendo:

P: El valor de la carga

L: La longitud total de la barra.

Al igual que en los casos anteriores, por medio del método de la doble integración obtenemos la flecha:

$$y = \frac{PLbx}{6EI}\left(1 - \frac{b^2}{L^2} - \frac{x^2}{L^2}\right) \qquad para \qquad 0 \le x \le a$$

$$y = \frac{PLa(L-x)}{6EI}\left(1 - \frac{a^2}{L^2} - \frac{(L-x)^2}{L^2}\right) \qquad para \qquad a \le x \le L$$

(68)

4.3.2.2. Estructuras reticuladas

Como ya se comentó con anterioridad en este mismo documento, en estas estructuras si se produce transmisión de esfuerzos entre nudos, por lo que, no solo cambiaremos los apoyos por empotramientos, si no que calcularemos el momento y la flecha a partir de las reacciones y la carga con dos premisas:

a) En las barras en las que nos encontremos con una carga aplicada, las reacciones no solo serán las cargas aplicadas, si no que se les sumará, a dichas fuerzas en los nodos, el producto de la matriz de rigidez en ejes globales de cada barra por el desplazamiento de los nodos inicial y final de dicha barra.

b) En las barras en las que no hay ninguna carga aplicada, sólo se tendrán en cuenta las fuerzas calculadas por el producto explicado mencionado.

$$\begin{bmatrix} H_A \\ V_A \\ M_A \\ H_B \\ V_B \\ M_B \end{bmatrix} = [K_{AB}]_{6x6} * \begin{Bmatrix} u_A \\ v_A \\ \theta_A \\ u_B \\ v_B \\ \theta_B \end{Bmatrix}$$

(69)

Las reacciones que se obtienen en dicho producto, estarán calculadas para ejes globales. Para el análisis que veremos en adelante, se aplicarán cargas en ejes locales, por lo que será necesario girarlas el ángulo correspondiente para cada barra.

Analizaremos la obtención de la flecha y el momento para cada uno de los tipos de cargas programados.

a) Carga uniformemente distribuida:

El momento flector en vigas cargadas con este tipo de carga en función de x responderá a la siguiente forma:

$$M_x = M_A - R_A \cdot x - P \cdot x \cdot \frac{x}{2} \quad (70)$$

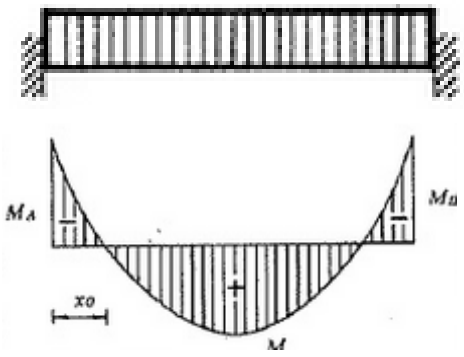


Figura 49. Momento flector viga biempotrada carga distribuida.

b) Carga triangularmente distribuida:

Haciendo un análisis idéntico al caso estudiado anteriormente, obtenemos:

$$M_x = M_A - V_A x - \frac{P x^3}{6L} \quad (71$$

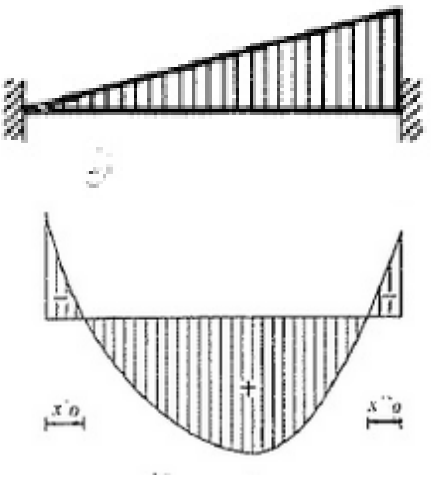


Figura 50. Momento flector viga biempotrada carga triangular.

Siendo para todas las ecuaciones:
L: La longitud de la barra

$$P = \frac{P_{max} * L}{2}$$

c) Carga puntual situada a una distancia 'a' del nodo inicial:

En este caso, la distancia 'a' será la existente a lo largo de la viga desde el nodo inicial hasta el punto de aplicación de la carga. Por el contrario, la distancia 'b' será lo que reste a la longitud total de la viga.
Con estas premisas, desarrollamos este caso de la misma forma que se hizo en el caso de carga distribuida uniforma mostrado anteriormente.

El momento flector de la parte anterior a la viga cambiará en cuanto a la posterior.

Siendo:

P: El valor de la carga

L: La longitud total de la barra.

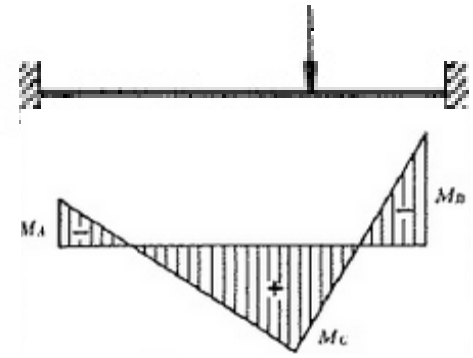


Figura 51. Momento flector viga biempotrada carga puntual.

$$M_x = M_A - R_A x \qquad para \quad 0 \leq x \leq a$$

$$M_x = M_A - R_A x - P(x - a) \qquad para \qquad a \leq x \leq L$$

(72)

4.4. Interfaz

Para una correcta apreciación de las mejoras realizadas, primeramente se hará una explicación del funcionamiento del UCMM, para más adelante explicar las mejoras llevadas a cabo que compondrán el UCMMv2.0

Debido a que las mejoras que han sido realizadas, únicamente afectan a la parte de resolución estática, mostraremos el funcionamiento de la interfaz de dicha parte.

4.4.1. Interfaz UCMM

Para la ejecución del programa se deberá escribir en la ventana de comandos de Matlab lo siguiente:

>> Menu_Principal

Que es el nombre del fichero que engloba y relaciona los distintos ficheros programados para calcular y obtener los valores de las frecuencias, modos propios, desplazamientos y fuerzas, de las distintas estructuras que el usuario se plantee resolver.

Aparecerá la siguiente ventana:

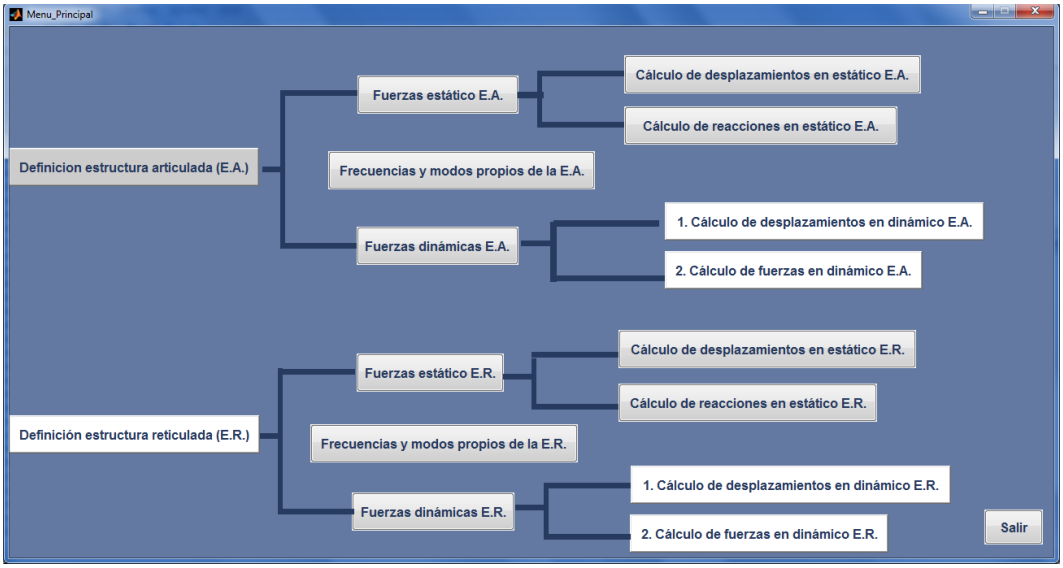


Figura 52. Menu principal UCMM.

En la ventana podremos distinguir cuatro tipos diferenciados de botones:

1. Los que nos permitirán elegir el tipo de estructura con el que trabajaremos y las propiedades de las mismas.

2. Los que nos permitirán elegir entre cargas estáticas o dinámicas

3. Los que nos muestran los modos propios de la estructura

4. Los que nos mostrarán los desplazamientos y reacciones calculadas.

4.4.1.1. Definición estructura

El primer paso que tenemos que llevar a cabo, es la definición de la estructura. Para ello, pulsaremos uno de los botones Definición estructura articulada (E.A) o Definición estructura reticulada (E.R) en función del tipo de estructura con el que queramos trabajar.

Independientemente del botón escogido, se nos abrirá una ventana como la siguiente:

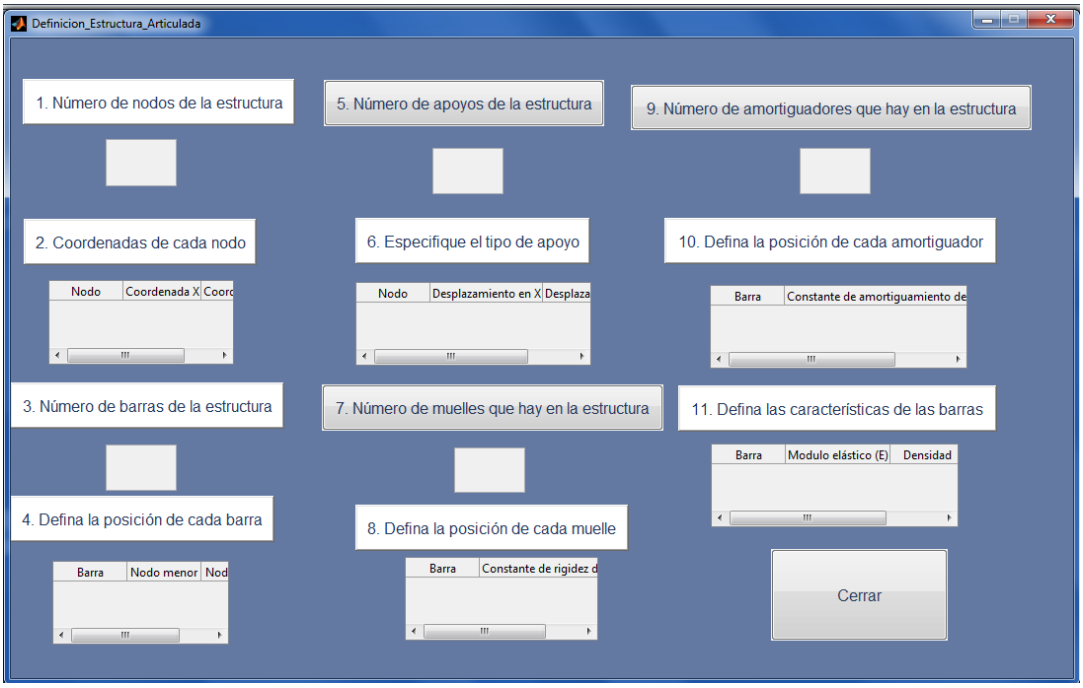


Figura 53. Ventana definición de estructuras.

Si observamos la ventana, aparecen 12 botones, 11 de ellos numerados, que explicaremos brevemente a continuación.

1. Número de nodos de la estructura: Abre una ventana que recoge el número de nodos con que estará la estructura formada.

2. Coordenadas de cada nodo: apretando este botón, se abrirá una ventana que recogerá la posición de cada uno de los nodos, esta se abrirá, se introducirán las coordenadas del primer nodo y al aceptar dichas coordenadas se volverá abrir para definir el siguiente nodo, y así sucesivamente hasta que estén todos los nodos definidos.

3. Número de barras de la estructura: De la misma manera que ocurría con los nodos, este botón recogerá el número de barras de las que está compuesta la estructura.

4. Defina la posición de cada barra: Exactamente igual que se recogían las posiciones de los nodos se recogerán los nodos iniciales y finales de todas las barras para que así queden completamente definidas.

5. Número de apoyos de la estructura: Botón que recoge el número de apoyos que tendrá la estructura.

6. Especifique el tipo de apoyo: Tantas veces como número de apoyos se indicaron en el comando anterior se abrirá una ventana que te pedirá el número de nodo en el que está el apoyo colocado y los movimientos que tendrá restringidos.

7. Defina el número de muelles: Botón que recoge el número de muelles que tendrá la estructura.

8. Defina la posición de cada muelle: Tantas veces como número de muelles hayas indicado en el apartado anterior, esta ventana se abrirá para recoger la barra en la que está el muelle colocado y el valor de su constante de rigidez K.

9. Defina el número de amortiguadores: Botón que recoge el número de muelles que tendrá la estructura.

10. Defina la posición de cada amortiguador: Tantas veces como número de amortiguadores hayas indicado en el apartado anterior, esta ventana se abrirá para recoger la barra en la que está el amortiguador colocado y el valor de su constante de amortiguamiento C.

11. Defina las características de las barras: al apretar este botón se abrirá una ventana diferente a los anteriores:

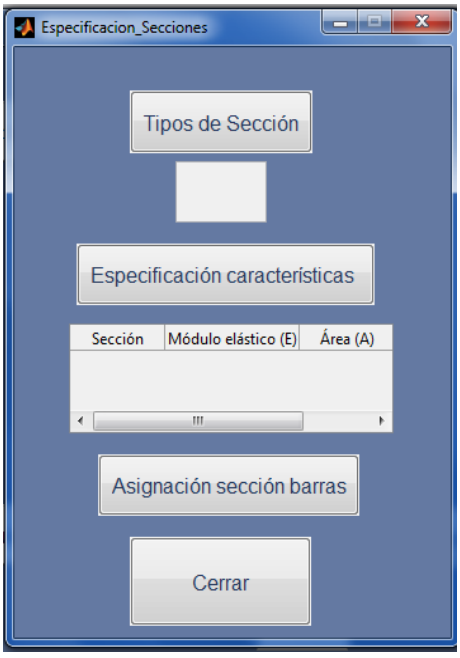


Figura 54. Ventana especificación seccion

En esta nueva ventana aparecerán 3 botones, empezando desde arriba, tendremos un primer botón que nos abrirá otra ventana para especificar el número de secciones que tendrá la estructura.

Apretando el siguiente botón, en función del número que le indiquemos en el apartado anterior, se abrirá una ventana tantas veces como el número indicado, en el que tendremos que especificar el Módulo elástico, el Área, la Densidad y el Momento de Inercia(este último sólo en el caso de reticuladas) de cada una de las secciones existentes.

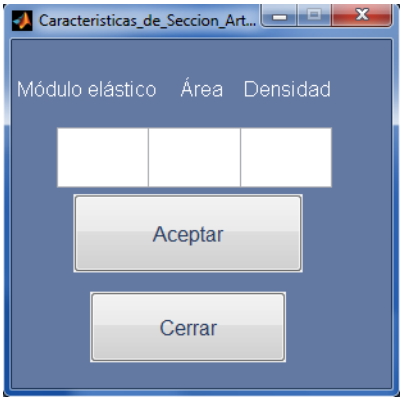


Figura 55. Características sección

El tercer botón que observamos, nos permitirá indicar, que sección tiene cada una de las barras que forman la estructura.

Llegados a este punto, la estructura ya quedaría totalmente definida. En la ventana de Definición_Estructura_Ariculada que hemos mostrado aparecen varias tablas debajo de los botones. En dichas tablas aparecerán los datos que hemos ido introduciendo, lo que nos permitirá revisar que todas las características estén correctamente definidas.

4.4.1.2. Definición de cargas

Ya tenemos la estructura totalmente definida, por lo que el siguiente paso es indicar las cargas que estarán aplicadas. Para ello pulsaremos uno de los siguientes botones:

- Fuerza Estático E.A.
- Fuerza Dinámicas E.A.
- Fuerza Estático E.R.
- Fuerza Dinámicas E.R.

Elegiremos el botón en función del tipo de estructura con el que estemos tratando y del tipo de carga que deseemos aplicar.

Como ya se ha comentado en apartados anteriormente, el programa inicial únicamente nos permite aplicar fuerzas en los nodos.

Cómo ya se comentó, profundizaremos en el caso estático, por lo que si por ejemplo pinchamos en el botón Fuerza Estático E.A. aparecerá la siguiente ventana:

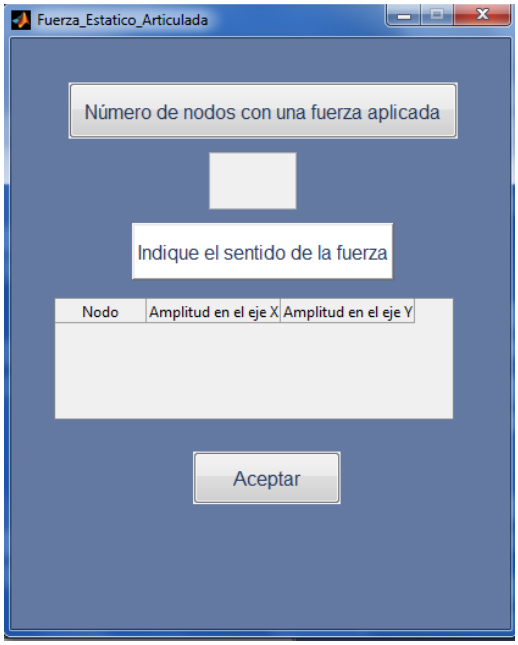


Figura 56. Ventana definición fuerzas UCMM.

Siguiendo la rutina de trabajo por la que está organizada el programa, primeramente introduciremos (con el primer botón empezando por la parte superior) el número de cargas que tiene la estructura.

Luego indicaremos las propiedades de las cargas que se podrán visualizar en la tabla.

En función de ese número introducido, se nos abrirá tantas veces como sea necesario la siguiente ventana que nos permitirá definir dichas cargas.

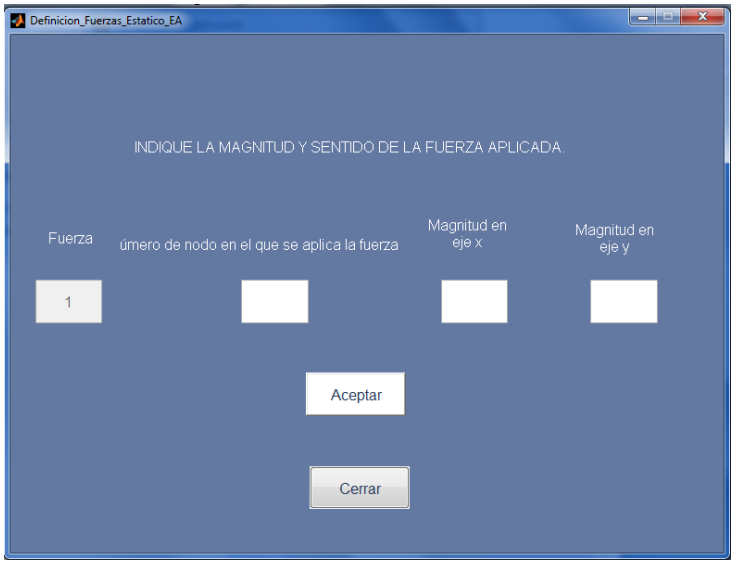


Figura 57. Propiedades fuerzas nodales.

De izquierda a derecha, el programa nos pedirá que le indiquemos el nodo en el que la aplicaremos la carga, la magnitud en el eje X y la magnitud en el eje Y. En el caso de fuerzas reticuladas, como los nudos también admiten momentos, habrá una cuarta ventana en la que podremos introducir un giro en el eje Z.

4.4.1.3. Visualización de resultados

Una vez tengamos la estructura totalmente definida y las cargas aplicadas, podremos observar los desplazamientos que se produjeron en los nodos no restringidos pulsando el botón, que aparecía en el menú principal, de Cálculo de desplazamientos en Estático E.A. o Cálculo de desplazamientos en Estático E.R. en función del tipo de estructura resuelta.

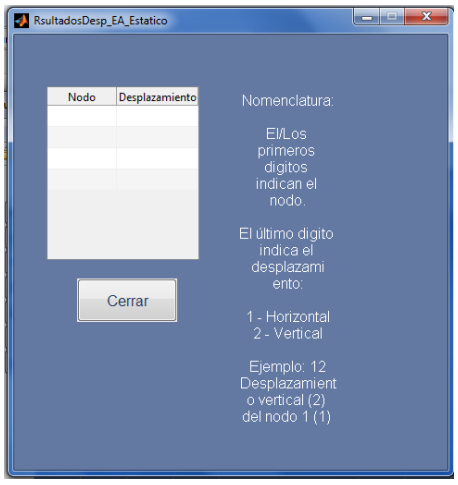


Figura 58.Visualización desplazamientos.

De la misma manera, el programa nos mostrará el vector de fuerzas y reacciones si pulsamos uno de los botones, que aparecían en el menú principal, de Cálculo de reacciones en estático E.A. o Cálculo de reacciones en estático E.R.

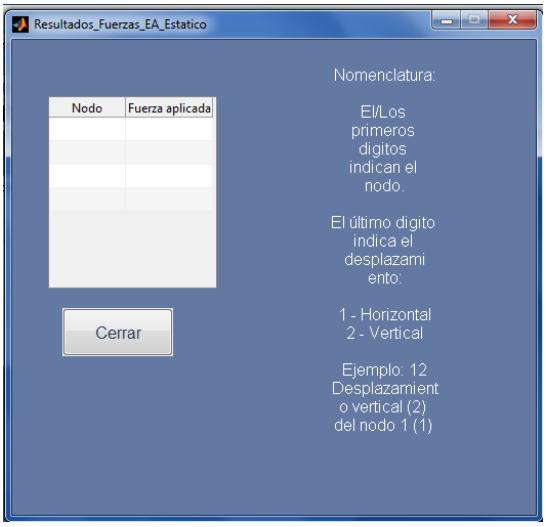


Figura 59. Visualización reacciones.

En las dos ventanas que nos proporcionan los mencionados resultados viene una pequeña leyenda de la manera en que vendrán expresados.

4.4.2. Interfaz UCMMv2.0

El menú principal del programa ha sido cambiado con comandos nuevos, a pesar de ello, la ejecución se llevará a cabo de la misma manera, es decir, escribiendo en la ventana de comandos:

>>Menu_Principal

Lo que nos abrirá la siguiente ventana:

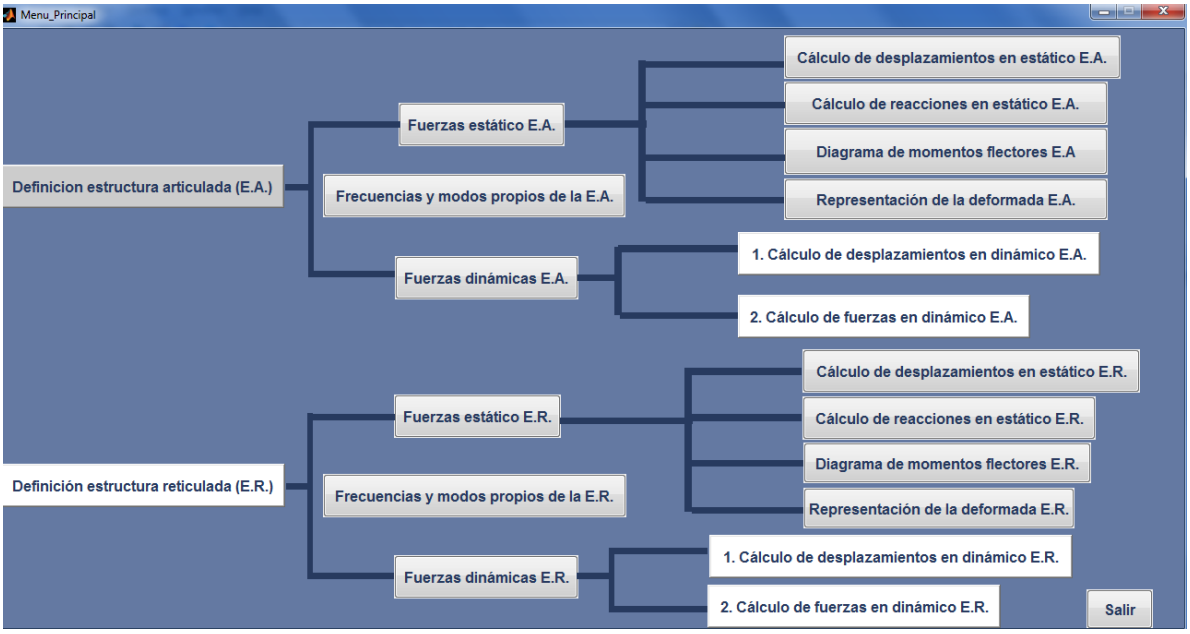


Figura 60. Menu principal UCMMv2.0.

En esta primera parte de la interfaz, observaremos dos botones nuevos por cada tipo de estructura: Diagrama de momentos flectores y Representación de la deformada.

La manera de proceder con el programa, no cambia en absoluto, teniendo que seguir los mismos pasos que se explicaron en la muestra del programa original (UCMM).

A continuación se enseñarán los cambios realizados en cada una de las partes.

4.4.2.1. Cambios en la descripción de la estructura

Esta parte es la que menores modificaciones ha sufrido.

En el caso de las estructuras articuladas, el programa para establecer la sección nos solicitaba que introdujéramos: área, módulo elástico y densidad, que eran los únicos datos que íbamos a necesitar para calcular desplazamientos y reacciones.

Una de las mejoras introducidas en el programa, es el cálculo de la deformada, por lo que para llevar a cabo dicho cálculo, es necesario introducir el momento de inercia de la sección, dato que hasta ahora, solo introducíamos en el caso de estructuras reticuladas.

Para solucionar esta incidencia, habilitamos un apartado más en la definición de la sección, que recoja este dato necesario para la resolución de la deformada.

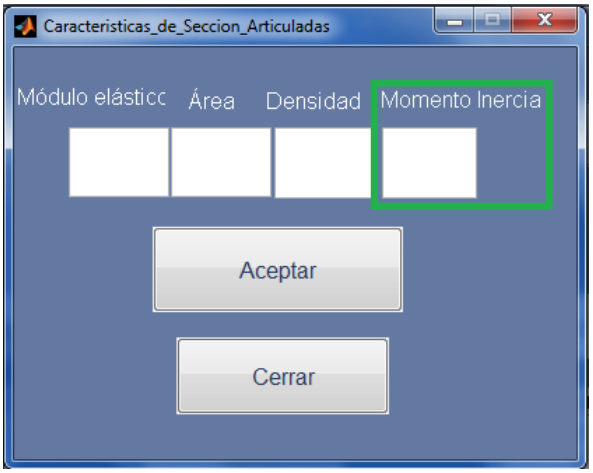


Figura 61. Características sección UCMMv2.0.

Observamos en la imagen el apartado extra que ha sido añadido para recoger el momento de inercia.

4.4.2.2. Cambios en la aplicación de cargas

Una de las mejoras de importancia que se le han realizado al programa es la de inserción de cargas no nodales tal y cómo se ha descrito en apartados anteriores.

En la nueva interfaz, si pulsamos los botones de definición de fuerzas estáticas aparecerá la siguiente ventana:

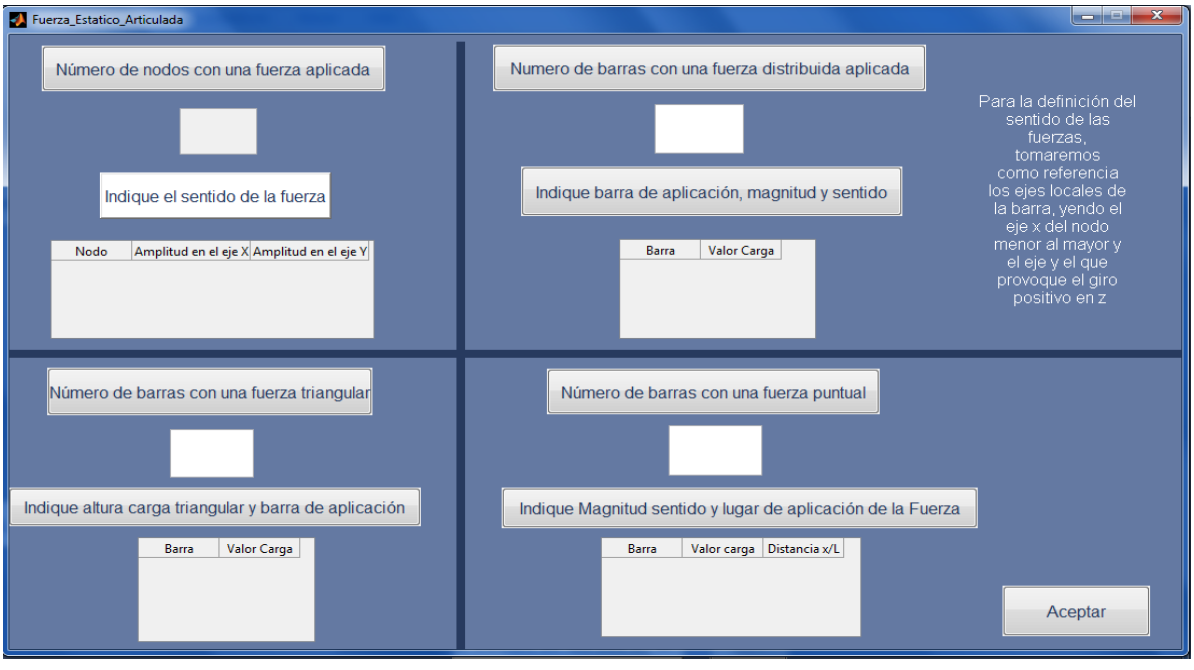


Figura 62. Definición fuerzas UCMMv2.0.

Ahora las posibilidades de meter distintos tipos de cargas han aumentado.

En la parte superior izquierda, continúa el apartado de inserción de fuerzas nodales, que trabaja tal y como lo hacía en el programa inicial.

4.4.2.2.1. Cargas distribuidas uniformemente

En la parte superior derecha aparece el apartado de fuerzas uniformemente distribuidas.

En primer lugar, pulsando el botón Número de barras con una fuerza distribuida aparecerá la siguiente ventana:

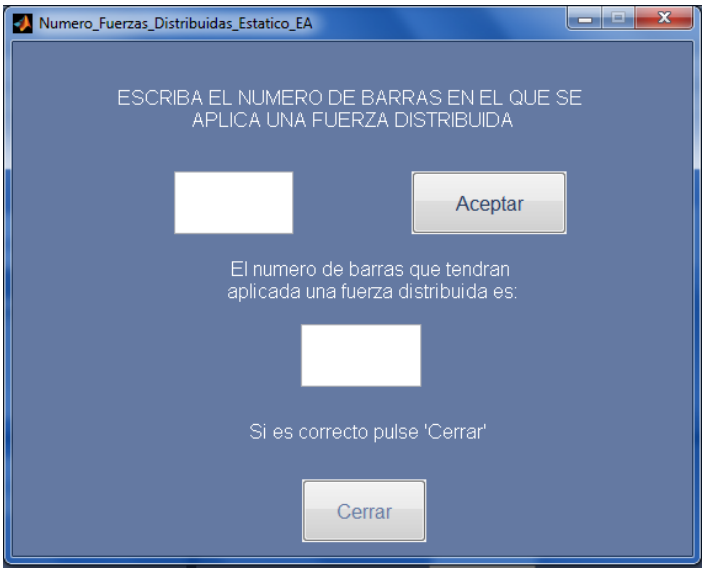


Figura 63. ventana número de fuerzas distribuidas.

En ella se recogerán el número de barras que estarán sometidas a una carga distribuida.

Para definir estas cargas distribuidas, lo haremos con el mismo proceso que se han ido describiendo los nodos, barras…etc.

Pulsando el botón Indique barra de aplicación, dirección y sentido aparecerá la siguiente ventana:

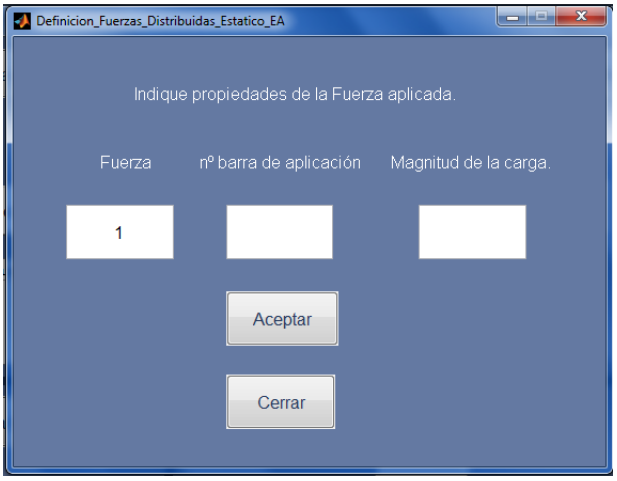


Figura 64. Propiedades cargas distribuidas.

En las dos cajas de texto editable que observamos introduciremos el número de barra que tendrá una carga distribuida y el valor de dicha carga. Si los datos son correctos daremos a Aceptar y esta ventana se volverá a abrir tantas veces como número de barras le fue indicado.

Una vez definidas estas cargas, aparecerán en la tabla inferior los datos que han sido introducidos, que nos servirá para comprobar si son correctos.

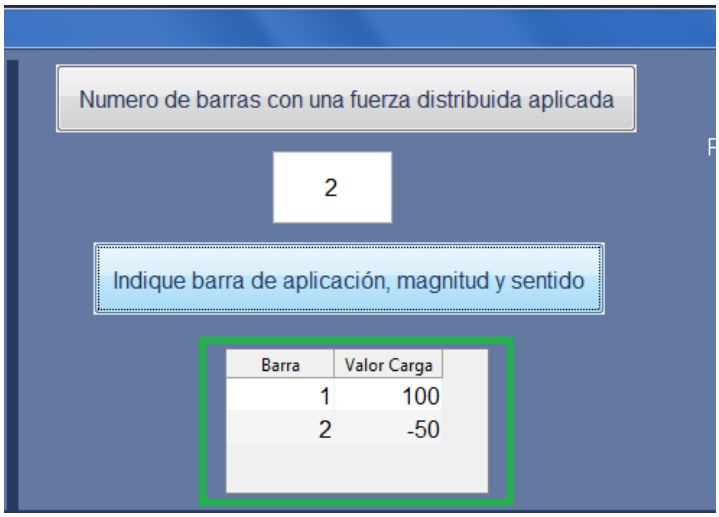


Figura 65. Comprobación datos insertados.

4.4.2.2.2. Cargas triangularmente distribuidas

En la parte inferior izquierda de la figura 61 , encontraremos el apartado de cargas triangulares. Este funcionará exactamente igual que el apartado anterior.

El botón superior Número de barras con una fuerza triangular, recogerá el número de barras que estarán cargadas con una fuerza de este tipo mediante la siguiente ventana:

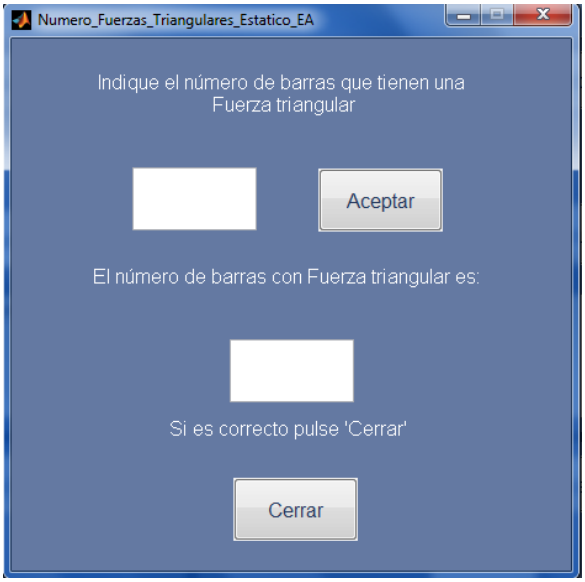


Figura 66.ventana número de fuerzas triangulares.

De forma análoga a la carga explicada anteriormente, tantas veces cómo número de cargas se le hayan indicado al programa, se abrirá la siguiente ventana:

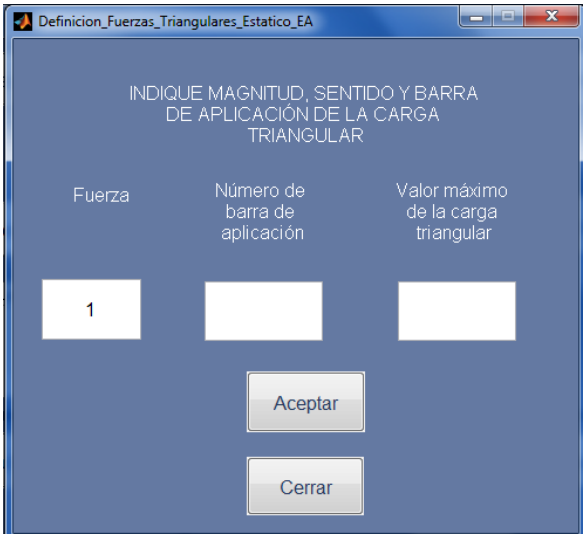


Figura 67. Propiedades cargas triangulares.

La cual recogerá la barra en que aplicaremos la fuerza y su valor máximo, es decir, la altura del triangulo que forma la carga.

En este caso también dispondremos de una tabla que mostrará los datos introducidos:

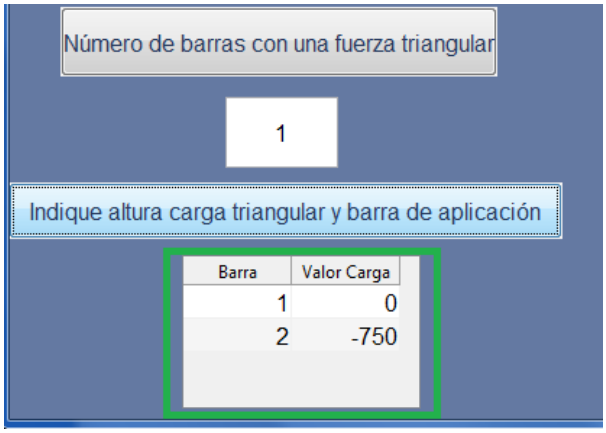


Figura 68.Comprobación cargas triangulares.

4.4.2.2.3. Carga puntual situada a una distancia a del nodo inicial

El último caso por describir es el situado en la esquina inferior derecha de la figura 61.

En este apartado, podremos introducir una carga puntual perpendicular a la barra colocada en cualquier punto de la misma.

Como se realizó con los demás tipos de cargas, se empezará introduciendo el número de fuerzas de este tipo que habrá en la estructura y luego definiremos sus propiedades.

Al pulsar el botón Número de barras con una fuerza puntual aparece la siguiente ventana:

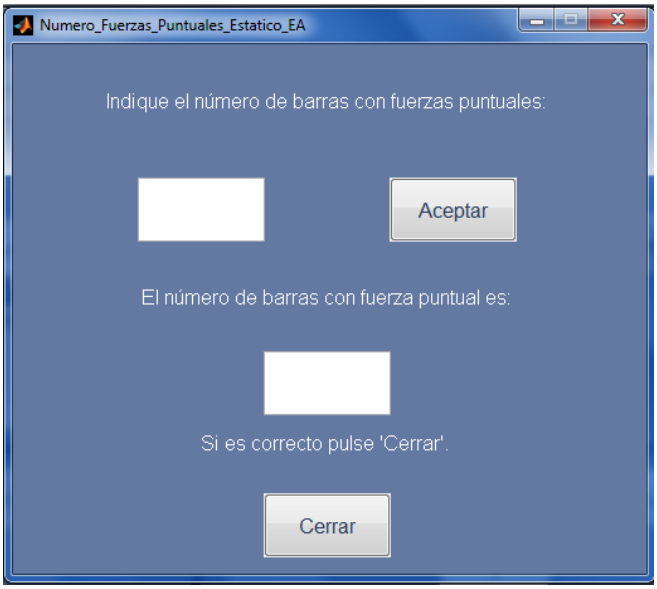


Figura 69.ventana número de fuerzas puntuales.

Una vez establecido el número de cargas, pulsando el botón Indique magnitud, sentido y lugar de aplicación de la carga aparecerá:

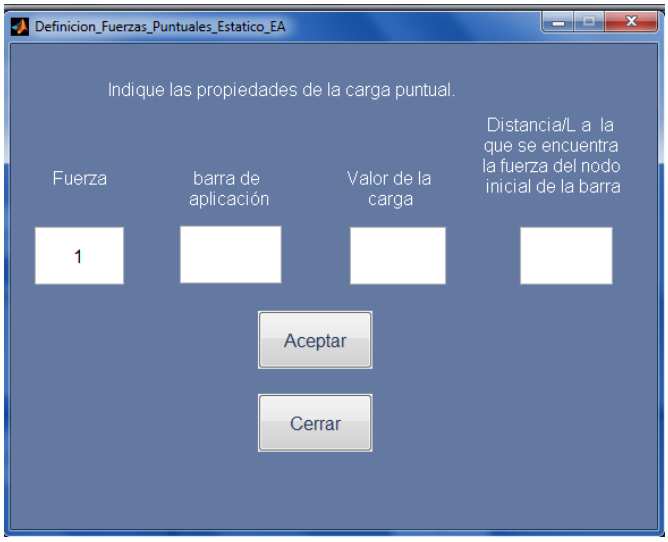


Figura 70. Propiedades cargas puntuales.

En este caso, tendremos que introducir tres datos. Como se ha ido haciendo hasta el momento en primer lugar indicaremos la barra en la que aplicaremos la carga y el valor de dicha carga. Para las cargas puntuales también tendremos que indicar su punto de aplicación.

De cara a una mayor comodidad, introduciremos el valor de la distancia/Longitud de la barra, de modo que no habrá que hacer un cálculo previo de la distancia exacta que hay desde el nodo inicial al punto de aplicación, cálculo algo engorroso cuando nos encontrásemos con barras inclinadas cierto alguno. Esto quiere decir que si por ejemplo queremos colocarla en el punto medio de la barra el valor que introduciríamos seria 0,5 sin necesidad de saber cuánto mide la barra o la distancia existente.

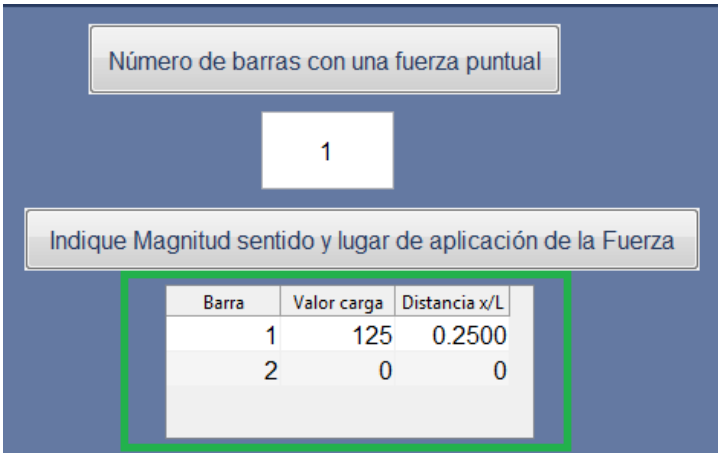


Figura 71. Comprobación cargas puntuales.

Si observamos la figura 61 , en la parte superior derecha, encontramos una leyenda explicativa del criterio de signos que seguiremos para las cargas.

Para una correcta elección del signo de la carga nos fijaremos en los ejes locales de la barra en la que aplicaremos la fuerza, estos irán el eje X positivo del nodo inicial al final de la barra, y el eje Y positivo el que genere un giro positivo en el eje Z.

Cuando nos encontremos con una estructura en la que no hay alguna de las cargas posibles, en esa carga es necesario decirle que el número de fuerzas es 0.

Una vez establecidos los cuatro tipos de fuerzas, al dar Aceptar, se generará, mediante los cálculos que ya se adelantaron en capítulos anteriores, el vector de cargas en los nodos que permitirá al programa calcular desplazamientos y reacciones.

4.4.2.3. Cambios en la visualización de resultados.

El programa actual está diseñado para calculas reacciones y desplazamientos. A la hora de analizar estructuras, en especial en estructuras reticuladas, uno de los apartados más importantes es el cálculo y representación de los esfuerzos a los que está sometida la estructura así como sus deformaciones

Por ello se consideró dar prioridad al cálculo de momentos flectores y de la deformada producida en la estructura.

En la interfaz aparecerán dos opciones nuevas en la visualización de resultados:

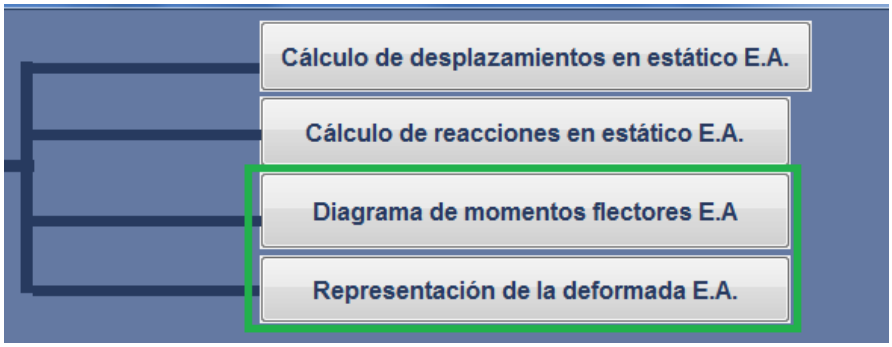


Figura 72. Cambios en el menú principal.

4.4.2.3.1 Diagrama de momentos flectores

Una vez establecidas las propiedades de la estructura y sus cargas, podremos visualizar los resultados obtenidos.

Si pulsamos el botón Diagrama de momentos flectores E.A. nos aparecerá la siguiente ventana:

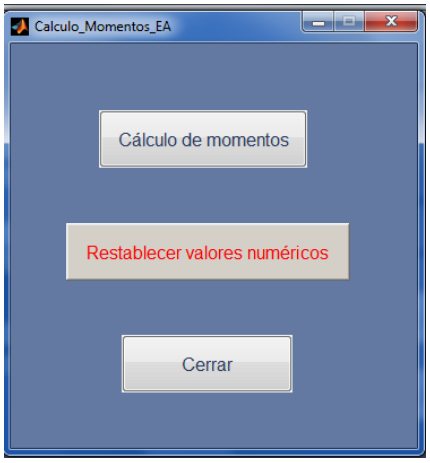


Figura 73. Ventana intermedia dibujo momentos.

Observamos tres botones:

- Cálculo de momentos: nos mostrará una ventana con el dibujo de la estructura y los momentos flectores producidos en ella, por otro lado nos indica el valor máximo de dicho momento en cada barra.

- Restablecer valores numéricos: Este botón estará marcado con letras rojas para que el usuario ponga un plus de atención en el, ya que pulsándolo limpia todos los datos introducidos al programa hasta el momento.

- Cerrar: Cierra la ventana mostrada.

La ventana en la que se mostrará el momento flector es la siguiente:

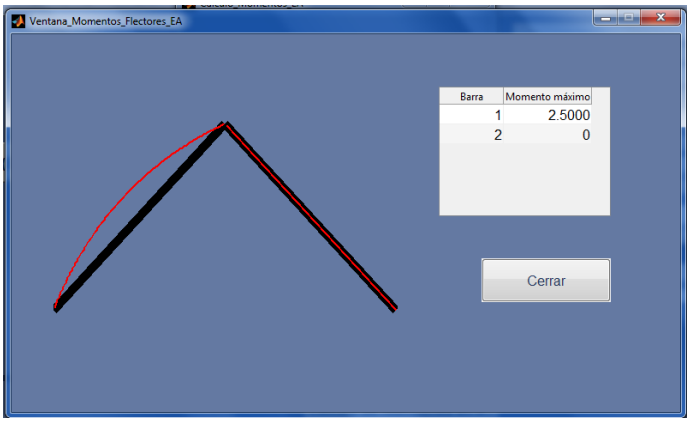


Figura 74. Ejemplo ventana momentos.

Se muestra una ventana con un ejemplo de dos barras y la aplicación de una carga uniforme en la barra de la izquierda.

En la parte de la izquierda de la ventana observamos el dibujo de la estructura con su ley de momentos flectores dibujada en rojo.

En la parte derecha se muestra una tabla con el valor máximo del momento en cada barra.

4.4.2.3.2. Representación de la deformada

Del mismo modo que pasaba con el momento flector actuaremos con la deformada.

Si pulsamos el botón Representación de la deformada E.A. aparece la siguiente ventana:

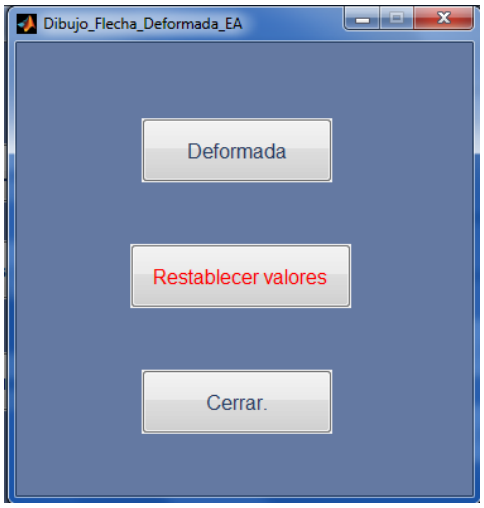


Figura 75. Ventana intermedia dibujo flecha.

En orden descendente empezando por arriba nos encontramos con los siguientes comandos:

- Deformada: Cuando pulsemos este botón, se nos mostrará una ventana con un dibujo de la estructura y su correspondiente deformada, así como los valores máximos para cada barra y los desplazamientos en los nodos. Esta acción tiene la peculiaridad de que previamente es necesario haber calculado los desplazamientos en los nodos ya que los necesita tener en cuenta para la representación de la deformada.

- Restablecer valores: Botón que trabaja de forma análoga al caso anterior, borrando todos los datos introducidos al programa.

- Cerrar: Cierra la ventana actual.

Para la visualización de la deformada se hará mediante la siguiente ventana:

En ella se mostrará el mismo ejemplo que en el caso anteriormente explicado.

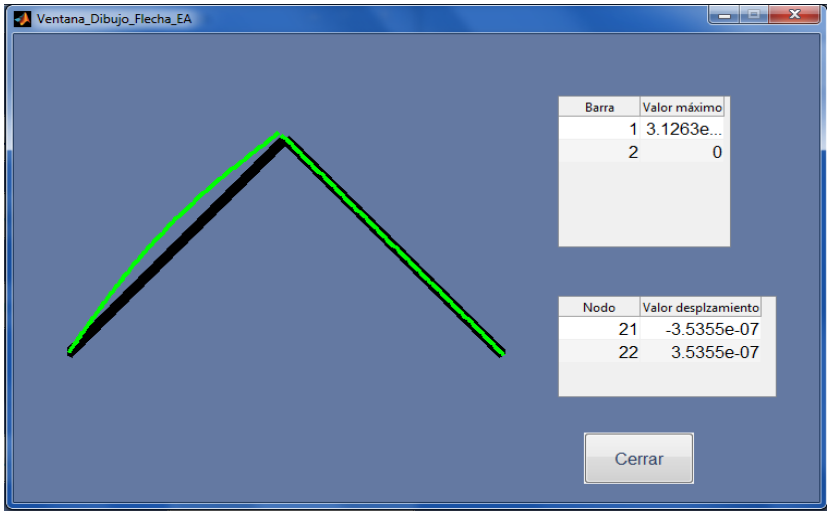


Figura 76. Ejemplo dibujo flecha.

En adición a la ventana que muestra el momento flector, se añade una tabla en la que se pueden ver los desplazamientos producidos en los nodos no restringidos.

4.5. Estructura de las mejoras

Siguiendo el patrón seguido por el UCMM, se han creado funciones (archivos *.m) que nos facilitarán el uso del programa y evitarán la introducción redundante de información así como una considerable reducción del cálculo computacional.

Algunos de estos ficheros serán modificaciones de los ficheros ya creados mientras que otros serán totalmente nuevos.

Dentro de estos ficheros, encontraremos dos tipos:

a) Ficheros *.m que nos crean la interfaz con la que trabajaremos.

b) Funciones que nos facilitarán el cálculo de las mejoras introducidas.

A continuación vemos los ficheros de los que se componen cada una de las partes.

Ficheros *.m que nos crean la interfaz con la que trabajaremos:

1. Menu_Principal.

2. Especificacion_Secciones_EA.

3. Especificacion_Secciones_ER.

4. Fuerza_Estatico_Articulada.

5. Fuerza_Estatico_Reticulada.

6. Numero_Fuerzas_Distribuidas_Estatico_EA.

7. Numero_Fuerzas_Distribuidas_Estatico_ER.

8. Definicion_Fuerzas_Distribuidas_Estatico_EA.

9. Definicion_Fuerzas_Distribuidas_Estatico_ER.

10. Numero_Fuerzas_Triangulares_Estatico_EA.

11. Numero_Fuerzas_Triangulares_Estatico_ER.

12. Definicion_Fuerzas_Triangulares _Estatico_EA.

13. Definicion_Fuerzas_Triangulares _Estatico_ER.

14. Numero_Fuerzas_Puntuales_Estatico_EA.

15. Numero_Fuerzas_Puntuales_Estatico_ER.

16. Definicion_Fuerzas_Puntuales_Estatico_EA.

17. Definicion_Fuerzas_Puntuales_Estatico_ER.

18. Calculo_Momentos_EA.

19. Ventana_Momentos_Flectores_EA.

20. Calculo_Momentos_ER.

21. Ventana_Momentos_Flectores_ER.

22. Dibujo_Flecha_Deformada_EA.

23. Ventana_Dibujo_Flecha_EA.

24. Dibujo_Flecha_Deformada_ER.

25. Ventana_Dibujo_Flecha_ER.

Funciones que nos facilitarán el cálculo de las mejoras introducidas:

23. Calculo_Reacciones_Barras_EA.

24. Calculo_Reacciones_Barras_ER.

25. Calculo_Momentos_Flectores_EA.

26. Calculo_Leyes_Momento_Flector_EA.

27. Calculo_Flecha_EA.

28. Calculo_Flecha_ER.

5. VALIDACIÓN DEL PROGRAMA

Para llevar a cabo la validación del programa, se llevará a cabo el desarrollo de varias estructuras, en las que por medio de un análisis estático, se hallarán las reacciones, desplazamientos en los nudos y los diagramas de momentos flectores y de la deformada.

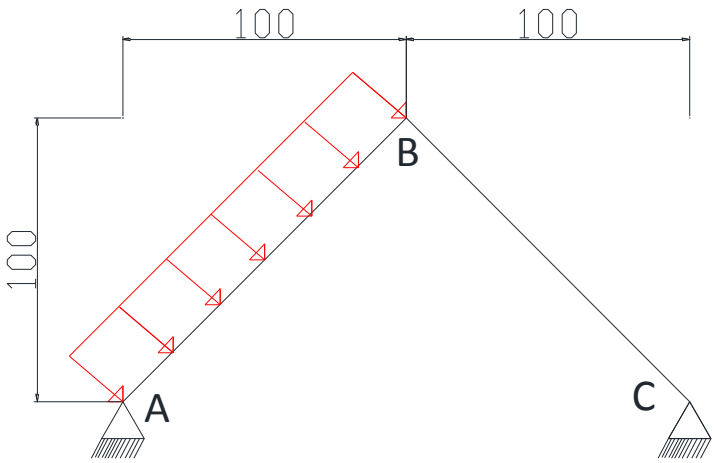
Los resultados obtenidos se compararán mediante los siguientes métodos:

- Un cálculo analítico.
- Cálculo mediante el programa Ed-Tridim.
- Cálculo mediante el programa UCMMv2.0.

Se realizará para estructuras articuladas y reticuladas, quedando así comprobado el buen funcionamiento del programa desarrollado.

5.1. Estructura articulada nº1

La primera estructura que desarrollaremos será para comprobar el buen funcionamiento de una carga uniformemente distribuida:



Las cotas están en cm.

Las propiedades de la misma se adjuntan en la siguiente tabla:

Longitud barras(cm)	Área(cm2)	Módulo Elástico(N/cm2)	Momento de Inercia(cm4)	Carga(N/cm)
141.42	100	200.000	833.33	1

5.1.1. Cálculo analítico

En primer lugar llevaremos a cabo el cálculo analítico.

El primer paso para ello es, como se habló en el capítulo de antecedentes, descomponer las fuerzas en las barras a fuerzas en los nodos.

Tenemos una carga uniformemente distribuida, por lo que para cada nodo tendremos la misma fuerza, siendo esta del mismo sentido y dirección y de valor:

$$R_A = R_B = \frac{P * L}{2} = \frac{1 * 141.42}{2} = 70.7\ N$$

(73)

De cara al análisis matricial, debemos descomponerla en ejes globales. El ángulo que forman las cargas respecto a los ejes globales es de 315º.

Por lo que tenemos:

$$R_{Ax} = R_A * cos\,315 = \ 50N$$

$$R_{Ay} = R_A * sin\,315 = \ -50N$$

(74)

Del mismo modo tendremos las mismas fuerzas nodales en el nodo B.

Siendo la barra nº 1 la situada a la izquierda de la estructura, entre el nodo A y el nodo B, y la barra nº2 la de la derecha, comprendida entre el nodo B y el nodo C, obtenemos el ángulo existente entre sus ejes locales y los globales.

Barra	Ángulo rotación entre ejes locales y globales (°)
1	45
2	135

Se tendrá que calcular la matriz de rigidez de los distintos elementos en coordenadas globales a partir de su expresión en locales y de las matrices de transformación de coordenadas.

Al tratarse de una estructura con nudos articulados, los movimientos permitidos en cada uno de ellos son únicamente los acortamientos o alargamientos en la dirección de los esfuerzos axiles (la propia dirección de la barra), por tanto la matriz de rigidez elemental de cada barra será de la forma:

$$\begin{bmatrix} \frac{EA}{L} & -\frac{EA}{L} \\ -\frac{EA}{L} & \frac{EA}{L} \end{bmatrix}$$

(75)

Matriz para la barra 1: α = 45°

$$K_1 = \frac{\sqrt{2}}{2}\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} * \frac{EA}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} * \frac{\sqrt{2}}{2}\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \frac{EA}{2L}\begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix}$$

(76)

Matriz para la barra 2: α=315°

$$K_2 = \frac{\sqrt{2}}{2}\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} * \frac{EA}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} * \frac{\sqrt{2}}{2}\begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} = \frac{EA}{2L}\begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

(77)

De manera que ensamblando ambas matrices, obtenemos la matriz global de la estructura:

$$K = \frac{EA}{2L}\begin{bmatrix} 1 & 1 & -1 & -1 & 0 & 0 \\ 1 & 1 & -1 & -1 & 0 & 0 \\ -1 & -1 & 2 & 0 & -1 & 1 \\ -1 & -1 & 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 & 1 & -1 \\ 0 & 0 & 1 & -1 & -1 & 1 \end{bmatrix}$$

(78)

Por lo que estableciendo el sistema de ecuaciones no reducidos:

$$\{F\} = [K] * \{X\}$$

$$\begin{bmatrix} H_A + 50 \\ V_A - 50 \\ 50 \\ -50 \\ H_C \\ V_C \end{bmatrix} = \frac{EA}{2L}\begin{bmatrix} 1 & 1 & -1 & -1 & 0 & 0 \\ 1 & 1 & -1 & -1 & 0 & 0 \\ -1 & -1 & 2 & 0 & -1 & 1 \\ -1 & -1 & 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 & 1 & -1 \\ 0 & 0 & 1 & -1 & -1 & 1 \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ \overrightarrow{u_B} \\ \overrightarrow{v_B} \\ 0 \\ 0 \end{bmatrix}$$

(79)

Estableciendo las ecuaciones de las filas 3 y 4 en las que aparecen los desplazamientos del nudo B, el único que no está restringido obtenemos:

$$50 = \frac{2EA}{2L}\,\overrightarrow{u_B} \quad \rightarrow \quad \overrightarrow{u_B} = 3{,}53\ 10^{-4}\ cm$$

$$-50 = \frac{2EA}{2L}\,\overrightarrow{v_B} \quad \rightarrow \quad \overrightarrow{v_B} = -3{,}53\ 10^{-4}\ cm$$

(80)

Una vez obtenidos los valores de los desplazamientos, el cálculo de las reacciones es inmediato, resolviendo las ecuaciones restantes del sistema global:

$$\begin{cases} H_A = -50\ N \\ V_A = 50\ N \\ H_C = -50\ N \\ V_C = 50\ N \end{cases}$$

Para realizar una comprobación del momento flector y de la flecha, se hallarán los valores máximos que se producirán en cada barra.

Barra 1: Dicha barra estará cargada con una carga distribuida de manera uniforme, por lo que sus valores máximos los encontraremos en el punto medio de la misma.

$$M_{max} = \frac{P*L^2}{8} = \frac{1*(141.42)^2}{8} = 2500\ Ncm$$

$$y = \frac{PL(L/2)^3}{12EI} - \frac{P(L/2)^4}{24EI} - \frac{P(L/2)L^3}{24EI} = 0.0314\ cm$$

(81)

Barra2: Esta barra no se encuentra cargada, por lo que el momento y la flecha en la misma serán nulos, ya que no se transmitirán esfuerzos entre nudos.

A continuación se muestra una tabla con todos los resultados obtenidos:

Desplazamientos(cm)	Reacciones y Fuerzas (N)	Momentos y Flechas
$u_A = 0$	$H_A = -50$	Barra 1
$v_A = 0$	$V_A = 50$	$M_{1max} = 2500\ Ncm$
$u_B = 3{,}53\ 10^{-4}$	-50	$y_{1max} = 0.0314\ cm$
$v_B = -3{,}53\ 10^{-4}$	50	Barra 2
$u_C = 0$	$H_C = -50$	$M_{2max} = 0\ Ncm$
$v_C = 0$	$V_C = 50$	$y_{2max} = 0\ cm$

5.1.2. Resolución con Ed-Tridim

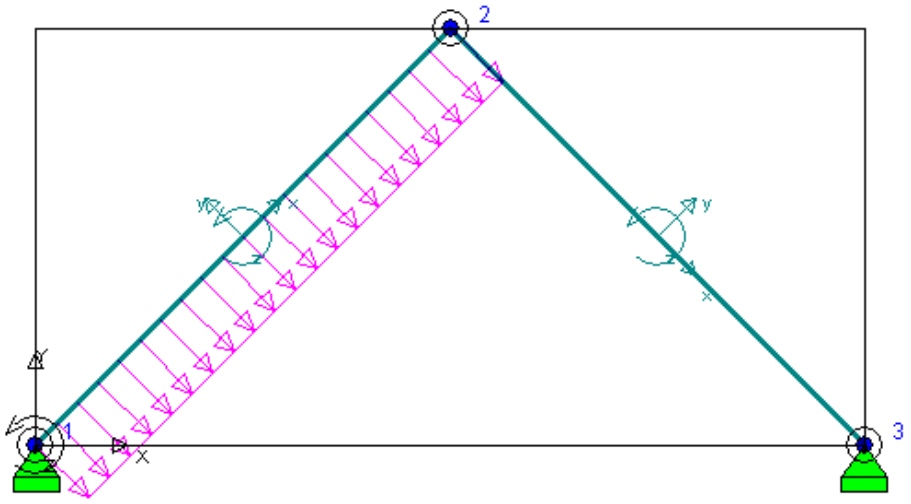


Figura 77. Estructura1 resuelta con Ed-Tridim.

En primer lugar se mostrarán las gráficas de los momentos y de la deformada de cara a una posterior comprobación con el UCMMv2.0 y con los valores máximos obtenidos en el cálculo analítico. Por otra parte se mostrarán la tabla con los desplazamientos y reacciones obtenidas.

Momento Flector:

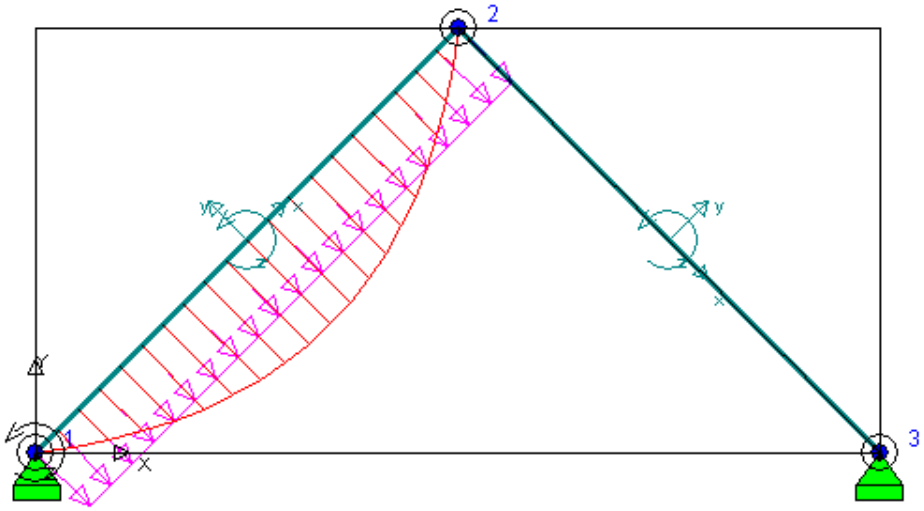


Figura 78. Momento flector estructura1.

Deformada:

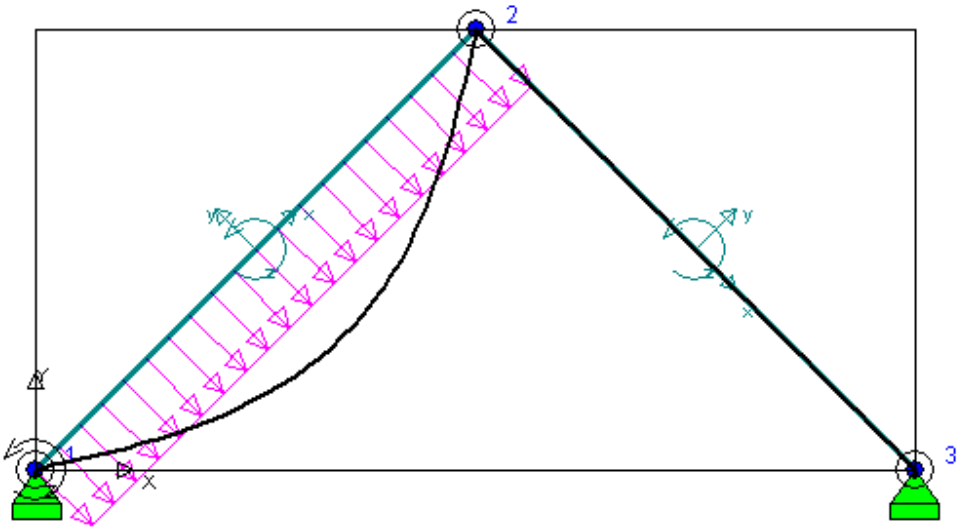


Figura 79. Deformada estructura1.

Desplazamientos(m)	Reacciones (N)	Momentos y Flechas
$u_1 = 0$	$H_A = -50$	Barra 1
$v_1 = 0$	$V_A = 50$	$M_{1max} = 25\ Nm$
$u_2 = 3{,}54\ 10^{-6}$		$y_{1max} = 0.000315\ m$
$v_2 = -3{,}54\ 10^{-6}$		Barra 2
$u_3 = 0$	$H_C = -50$	$M_{2max} = 0\ Nm$
$v_3 = 0$	$V_C = 50$	$y_{2max} = 0\ cm$

5.1.3. Resolución con UCMMv2.0

Diagrama de momentos:

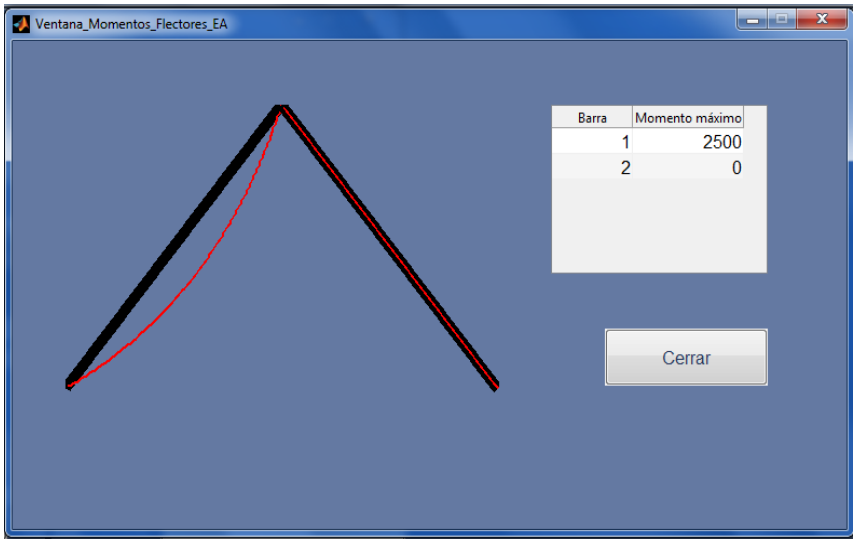


Figura 80. Momentos estructura1 por UCMMv2.0.

Deformada:

En la representación de la deformada mediante el UCMMv2.0 se incluye el desplazamiento producido en los nodos, escalado de forma que el usuario pueda apreciarlo.

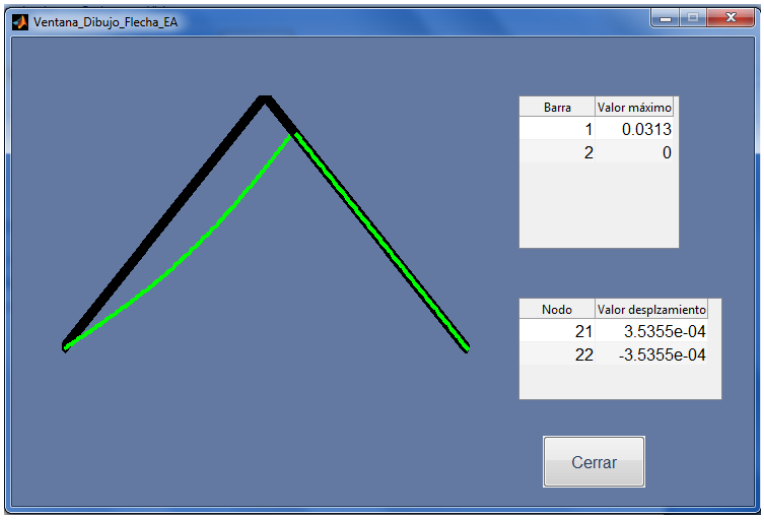


Figura 81. Deformada estructura1 por UCMMv2.0.

Desplazamientos(cm)	Reacciones y Fuerzas (N)	Momentos y Flechas
$u_1 = 0$	$H_1 = -50$	Barra 1
$v_1 = 0$	$V_1 = 50$	$M_{1max} = 2500\ Ncm$
$u_2 = 3{,}5355\ 10^{-4}$	-50	$y_{1max} = 0.0313\ cm$
$v_2 = -3{,}5355\ 10^{-4}$	50	Barra 2
$u_3 = 0$	$H_3 = -50$	$M_{2max} = 0\ Nm$
$v_3 = 0$	$V_3 = 50$	$y_{2max} = 0\ cm$

A continuación se mostrarán las comparativas de los resultados obtenidos en los tres modos de resolución:

Tabla de desplazamientos:

UCMMv2.0(cm)	Ed-Tridim(m)	Cálculo analítico(cm)
$u_1 = 0$	$u_1 = 0$	$u_A = 0$
$v_1 = 0$	$v_1 = 0$	$v_A = 0$
$u_2 = 3{,}5355\ 10^{-4}$	$u_2 = 3{,}54\ 10^{-6}$	$u_B = 3{,}53\ 10^{-4}$
$v_2 = -3{,}5355\ 10^{-4}$	$v_2 = -3{,}54\ 10^{-6}$	$v_B = -3{,}53\ 10^{-4}$
$u_3 = 0$	$u_3 = 0$	$u_C = 0$
$v_3 = 0$	$v_3 = 0$	$v_C = 0$

Los desplazamientos obtenidos en esta primera estructura son muy similares, discrepando en valores muy pequeños. En la tabla varían las potencias en función de las unidades.

Tabla de reacciones:

UCMMv2.0(N)	Ed-Tridim(N)	Cálculo analítico(N)
$H_1 = -50$	$H_A = -50$	$H_A = -50$
$V_1 = 50$	$V_A = 50$	$V_A = 50$
-50		-50
50		50
$H_3 = -50$	$H_C = -50$	$H_C = -50$
$V_3 = 50$	$V_C = 50$	$V_C = 50$

Tabla de valores máximos de la deformada y el diagrama de momentos:

UCMMv2.0	Ed-Tridim	Cálculo analítico
Barra 1	**Barra 1**	**Barra 1**
$M_{1max} = 2500\ Ncm$	$M_{1max} = 25\ Nm$	$M_{1max} = 2500\ Ncm$
$y_{1max} = 0.0313\ cm$	$y_{1max} = 0.000315\ m$	$y_{1max} = 0.0314\ cm$
Barra 2	**Barra 2**	**Barra 2**
$M_{2max} = 0\ Nm$	$M_{2max} = 0\ Nm$	$M_{2max} = 0\ Ncm$
$y_{2max} = 0\ cm$	$y_{2max} = 0\ cm$	$y_{2max} = 0\ cm$

Todos los valores obtenidos difieren muy poco entre ellos, considerando que, de momento, para esta primera estructura, el programa es totalmente válido.

5.2. Estructura articulada nº2

5.2.1. Resolución con Ed-Tridim

En la siguiente estructura, comprobaremos el buen funcionamiento para una carga triangular:

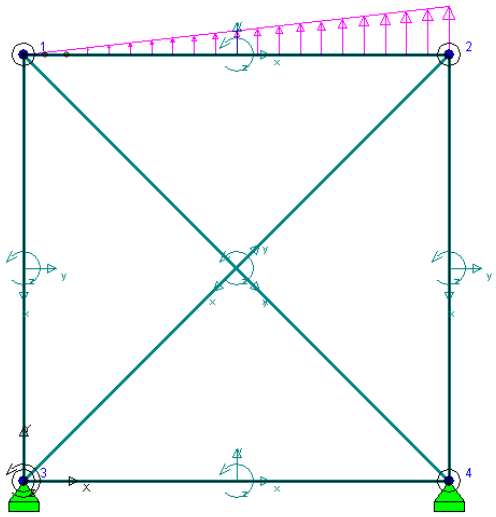


Figura 82. Estructura2.

Longitud barras exteriores(cm)	Área(cm2)	Módulo Elástico(N/cm2)	Momento de Inercia(cm4)	Carga(N/cm)
$L_{12} = L_{24} = 400$	100	200.000	833.33	1

Momento Flector:

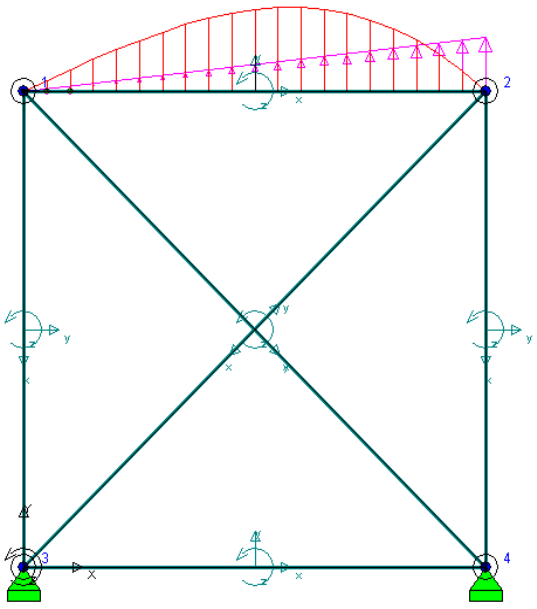


Figura 83.Diagrama de momentos por Ed-Tridim.

Deformada:

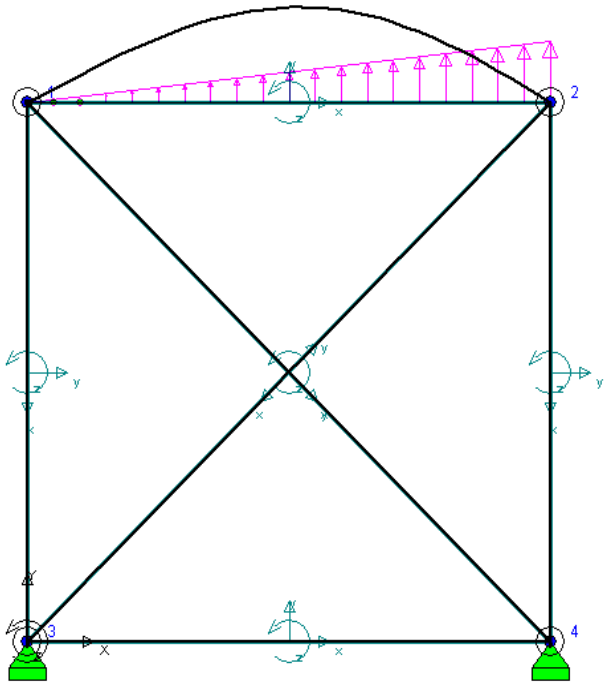


Figura 84.Deformad por Ed-Tridim.

Tabla de desplazamientos, reacciones y valores máximos:

En la única barra que aparecerán momentos será en la barra en la que se encuentra la barra aplicada, es decir la barra que une el nodo 1 y el nodo 2.

Desplazamientos(m)	Reacciones y Fuerzas (N)	Momentos y Flechas
$u_1 = -4{,}36\ 10^{-6}$		Barra 1
$v_1 = 8{,}71\ 10^{-6}$		$M_{1max} = 103\ Nm$
$u_2 = -8.98\ 10^{-6}$		$y_{1max} = 0.01\ m$
$v_2 = 2.2\ 10^{-5}$		
$u_3 = 0$	$H_3 = -23.1$	
$v_3 = 0$	$V_3 = -66.7$	
$u_4 = 0$	$H_4 = -23.1$	
$v_4 = 0$	$V_4 = -133$	

5.2.2. Resolución con UCMMv2.0

Momento Flector:

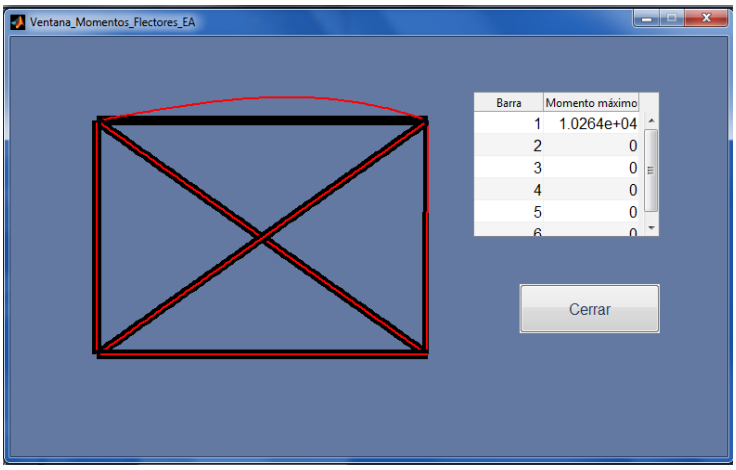


Figura 85. diagrama de momentos por UCMMv2.0.

Deformada:

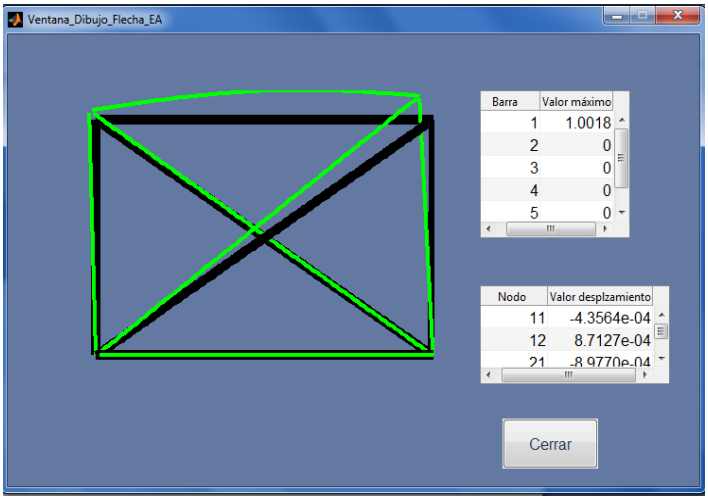


Figura 86.Deformada por UCMMv2.0.

Tabla desplazamientos, reacciones y valores máximos:

Desplazamientos(cm)	Reacciones y Fuerzas (N)	Momentos y Flechas
$u_1 = -4{,}3564\ 10^{-4}$	0	Barra 1
$v_1 = 8{,}7127\ 10^{-4}$	66.67	$M_{1max} = 10264\ Ncm$
$u_2 = -8.977\ 10^{-4}$	$7.10\ 10^{-15}$	$y_{1max} = 1.0018\ cm$
$v_2 = 2.2\ 10^{-3}$	133.333	
$u_3 = 0$	$H_3 = -23.1031$	
$v_3 = 0$	$V_3 = -66.667$	
$u_4 = 0$	$H_4 = 23.1031$	
$v_4 = 0$	$V_4 = -133.33$	

Comparamos los resultados obtenidos con ambos programas:

Tabla desplazamientos:

UCMMv2.0(cm)	Ed-Tridim(m)
$u_1 = -4{,}3564\ 10^{-4}$	$u_1 = -4{,}36\ 10^{-6}$
$v_1 = 8{,}7127\ 10^{-4}$	$v_1 = 8{,}71\ 10^{-6}$
$u_2 = -8.977\ 10^{-4}$	$u_2 = -8.98\ 10^{-6}$
$v_2 = 2.2\ 10^{-3}$	$v_2 = 2.2\ 10^{-5}$
$u_3 = 0$	$u_3 = 0$
$v_3 = 0$	$v_3 = 0$
$u_4 = 0$	$u_4 = 0$
$v_4 = 0$	$v_4 = 0$

Tabla reacciones:

UCMMv2.0(N)	Ed-Tridim(N)
0	
66.67	
$7.10\ 10^{-15}$	
133.333	
$H_3 = -23.1031$	$H_3 = -23.1$
$V_3 = -66.667$	$V_3 = -66.7$
$H_4 = 23.1031$	$H_4 = -23.1$
$V_4 = -133.33$	$V_4 = -133$

Tabla valores máximos flecha y momento flector:

UCMMv2.0	Ed-Tridim
Barra 1	Barra 1
$M_{1max} = 10264\ Ncm$	$M_{1max} = 103\ Nm$
$y_{1max} = 1.0018\ cm$	$y_{1max} = 0.01\ m$

En este caso, los resultados tanto de reacciones como de desplazamientos coinciden con variaciones muy pequeñas.

En el caso de la deformada, varía ligeramente más debido a que en el programa UCMMv2.0 se tienen en cuenta los desplazamientos nodales. A pesar de ello, los valores máximos de la flecha y del momento también son similares.

5.3. Estructura articulada nº3

5.3.1. Resolución con Ed-Tridim

En esta estructura comprobaremos el correcto funcionamiento de la aplicación de una carga puntual en la barra.

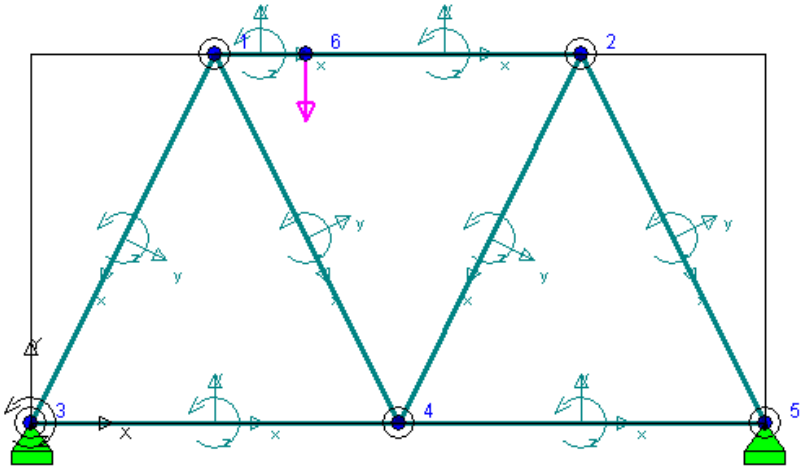


Figura 87. Estructura3.

Longitud barras (cm)	Altura (cm)	Área (cm2)	Módulo Elástico(N/cm2)	Momento de Inercia(cm4)	Carga (N)
$L_{12} = L_{34} = L_{45} = 200$	200	100	200.000	833.33	100

Momento Flector:

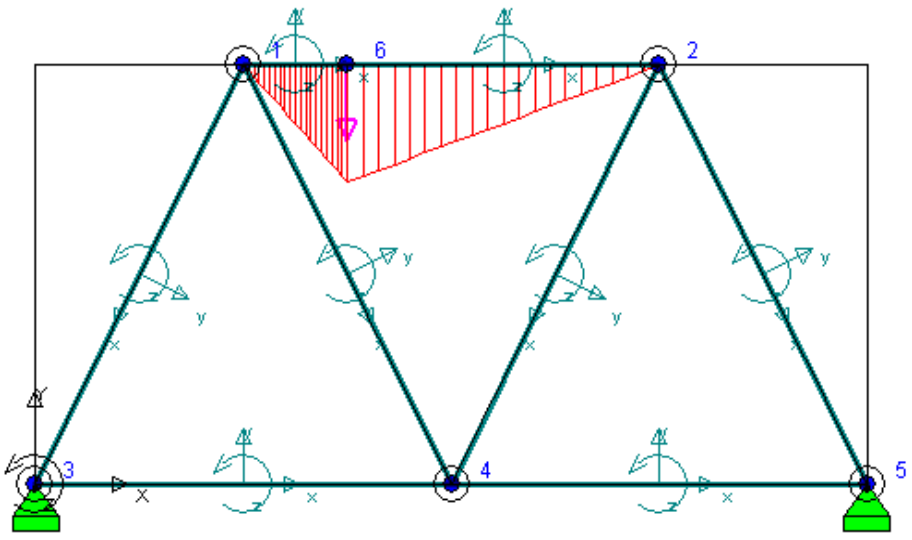


Figura 88. Diagrama de momentos por Ed-Tridim.

Deformada:

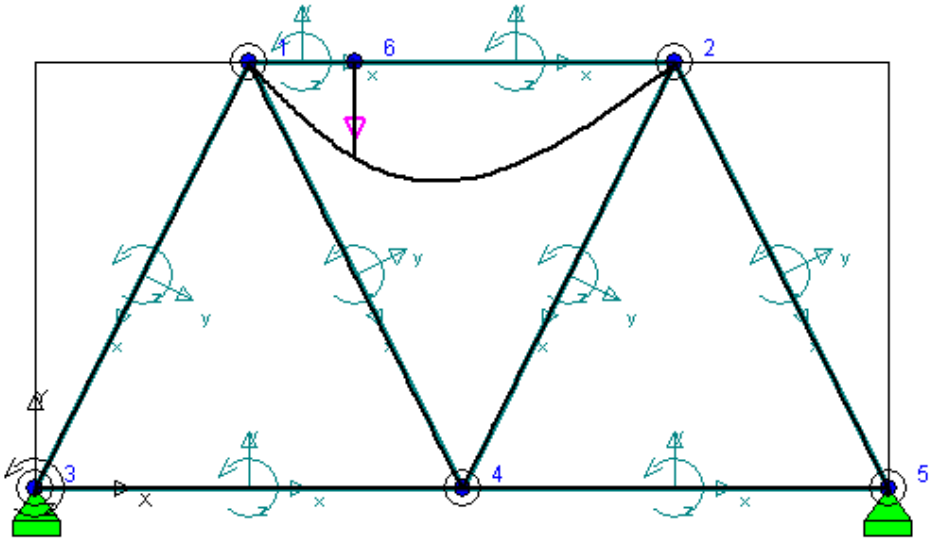


Figura 89. Deformada por Ed-Tridim.

Tabla de desplazamientos, reacciones y valores máximos:

La barra cargada será la barra 1, que une el nodo 1 y el nodo 2. En el programa Ed-Tridim, es necesaria la colocación de un nodo rígido en el lugar donde se aplicará la carga, este será el nodo 6.

Desplazamientos(m)	Reacciones (N)	Momentos y Flechas
$u_1 = 1{,}56\ 10^{-6}$		Barra 1
$v_1 = -9{,}52\ 10^{-6}$		$M_{1max} = 37.5Nm$
$u_2 = -9{,}37\ 10^{-7}$		$y_{1max} = 0.000707\ m$
$v_2 = -5{,}71\ 10^{-6}$		
$u_3 = 0$	$H_3 = 25$	
$v_3 = 0$	$V_3 = 62.5$	
$u_2 = 6{,}25\ 10^{-7}$		
$v_2 = -8{,}24\ 10^{-6}$		
$u_4 = 0$	$H_5 = -25$	
$v_4 = 0$	$V_5 = 37.5$	

5.3.2. Resolución con UCMMv2.0.

Momento Flector:

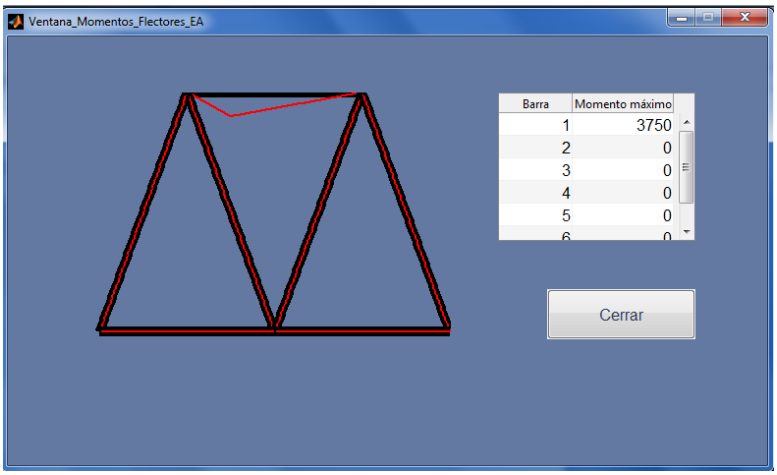


Figura 90. Diagrama de momentos por UCMMv2.0.

Deformada:

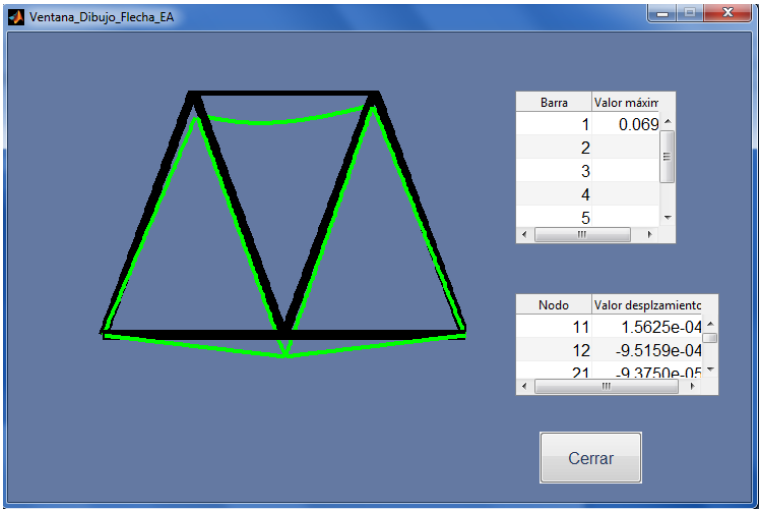


Figura 91. Deformada por UCMMv2.0.

Tabla de desplazamientos, reacciones y valores máximos:

Desplazamientos(cm)	Reacciones (N)	Momentos y Flechas
$u_1 = 1{,}5625\ 10^{-4}$	0	Barra 1
$v_1 = -9{,}5159\ 10^{-4}$	-75	$M_{1max} = 3750\ Ncm$
$u_2 = -9{,}375\ 10^{-5}$	0	$y_{1max} = 0.069\ cm$
$v_2 = -5{,}7095\ 10^{-4}$	-25	
$u_3 = 0$	$H_3 = 25$	
$v_3 = 0$	$V_3 = 62.5$	
$u_2 = 6{,}25\ 10^{-5}$	0	
$v_2 = -8{,}2377\ 10^{-4}$	0	
$u_4 = 0$	$H_5 = -25$	
$v_4 = 0$	$V_5 = 37.5$	

Comparamos los resultados obtenidos en ambos programas:

Tabla de desplazamientos:

UCMMv2.0(cm)	Ed-Tridim(m)
$u_1 = 1{,}5625\ 10^{-4}$	$u_1 = 1{,}56\ 10^{-6}$
$v_1 = -9{,}5159\ 10^{-4}$	$v_1 = -9{,}52\ 10^{-6}$
$u_2 = -9{,}375\ 10^{-5}$	$u_2 = -9{,}37\ 10^{-7}$
$v_2 = -5{,}7095\ 10^{-4}$	$v_2 = -5{,}71\ 10^{-6}$
$u_3 = 0$	$u_3 = 0$
$v_3 = 0$	$v_3 = 0$
$u_2 = 6{,}25\ 10^{-5}$	$u_2 = 6{,}25\ 10^{-7}$
$v_2 = -8{,}2377\ 10^{-4}$	$v_2 = -8{,}24\ 10^{-6}$
$u_4 = 0$	$u_4 = 0$
$v_4 = 0$	$v_4 = 0$

Tabla de reacciones:

UCMMv2.0(N)	Ed-Tridim(N)
0	
-75	
0	
-25	
$H_3 = 25$	$H_3 = 25$
$V_3 = 62.5$	$V_3 = 62.5$
0	
0	
$H_5 = -25$	$H_5 = -25$
$V_5 = 37.5$	$V_5 = 37.5$

Tabla de valores máximos:

UCMMv2.0	Ed-Tridim
Barra 1	Barra 1
$M_{1max} = 3750\ Ncm$	$M_{1max} = 37.5Nm$
$y_{1max} = 0.069\ cm$	$y_{1max} = 0.000707\ m$

Con este tercer ejemplo, en el que también obtenemos los mismos valores con ambos métodos, queda validado el programa en cuanto a la parte de estructuras articuladas.

5.4. Estructura Reticulada nº1

La primera estructura que desarrollaremos será para comprobar el buen funcionamiento de una carga uniformemente distribuida:

Longitud barras (cm)	Área (cm2)	Módulo Elástico(N/cm2)	Momento de Inercia(cm4)	Cargas (N)
$L_{12} = L_{23} = L_{34} = 200$	100	200.000	833.33	100

5.4.1. Resolución con Ed-Tridim

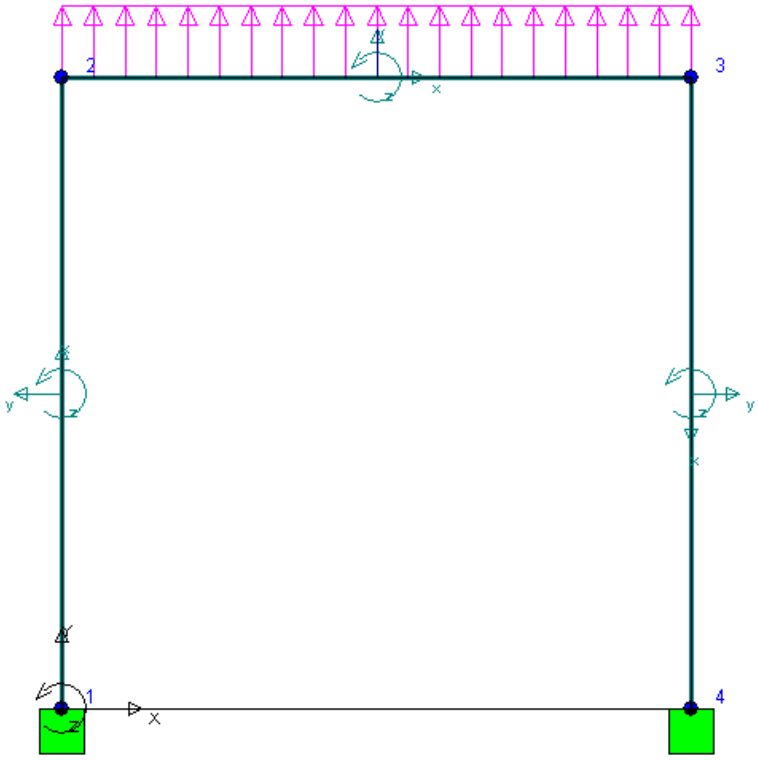


Figura 92.Estructura 4.

Momento Flector:

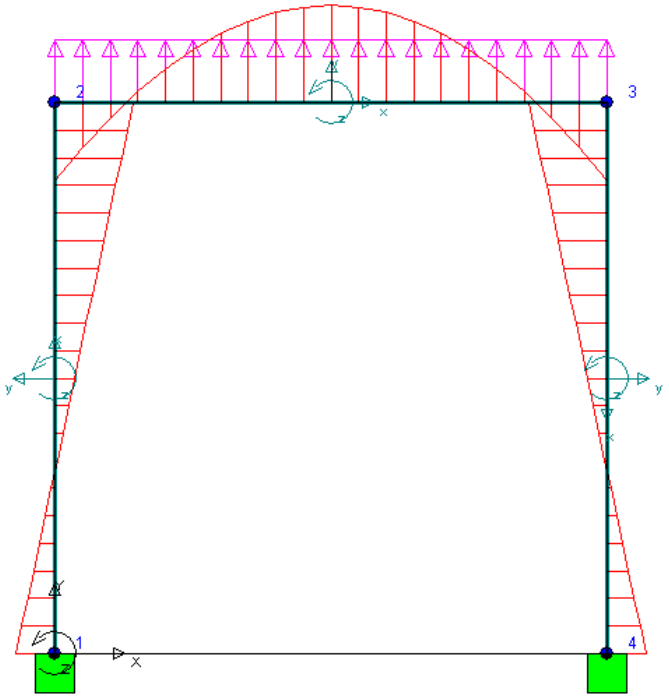


Figura 93. Diagrama de momentos por Ed-Tridim.

Tabla de desplazamientos, reacciones y valores máximos:

Desplazamientos(m)(rad)	Reacciones (N)(Nm)	Momentos y Flechas
$u_1 = 0$	$H_1 = -16.7$	Barra 1
$v_1 = 0$	$V_1 = -100$	$M_{1max} = 22.2\ Nm$
$\Theta_1 = 0$	$M_1 = 11.1$	$y_{1max} = 0.00019\ m$
$u_2 = -8.33\ 10^{-7}$		Barra 2
$v_2 = 1\ 10^{-5}$		$M_{2max} = 27.8\ Nm$
$\Theta_2 = 0.000667$		$y_{2max} = 0.0005\ m$
$u_3 = 8.33\ 10^{-7}$		Barra 3
$v_3 = 1\ 10^{-5}$		$M_{3max} = 22.2\ Nm$
$\Theta_3 = -0.000667$		$y_{3max} = 0.00019\ m$
$u_4 = 0$	$H_4 = 16.7$	
$v_4 = 0$	$V_4 = 100$	
$\Theta_4 = 0$	$M_4 = 11.1$	

5.4.2. Resolución con UCMMv2.0

Diagrama de Momentos:

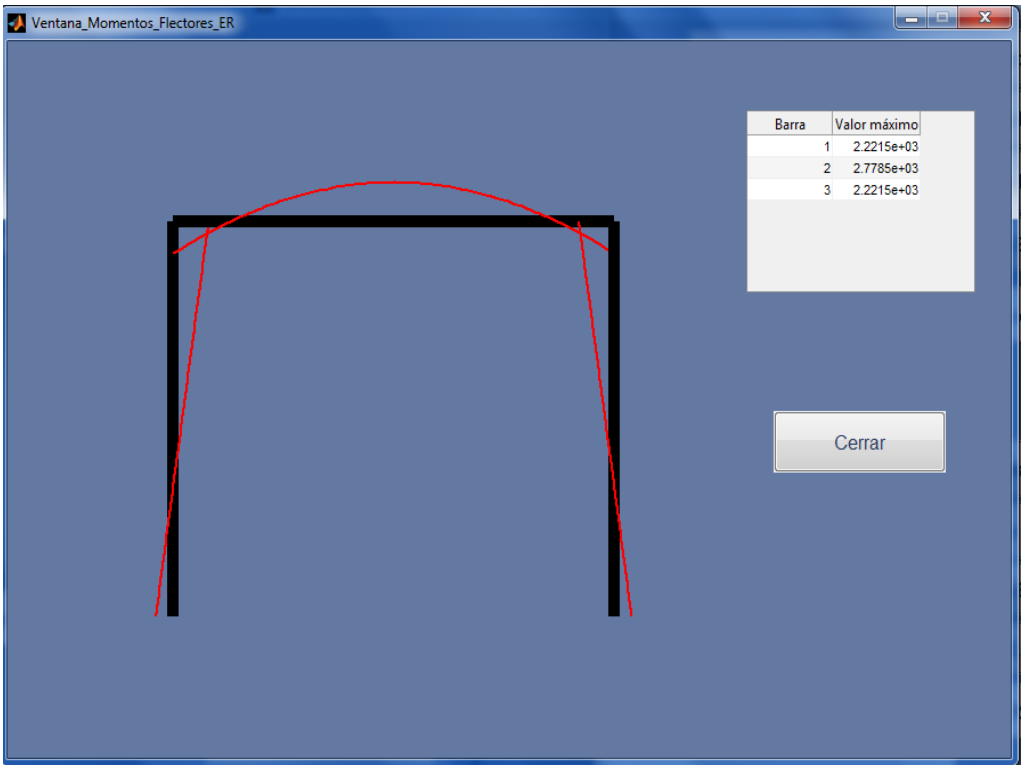


Figura 94. Diagrama de momentos estructura4 por UCMMv2.0.

Tabla de desplazamientos, reacciones y valores máximos:

Desplazamientos(cm)(rad)	Reacciones (N)(Ncm)	Momentos y Flechas
$u_1 = 0$	$H_1 = -16.7$	Barra 1
$v_1 = 0$	$V_1 = -100$	$M_{1max} = 2221\ Ncm$
$\Theta_1 = 0$	$M_1 = 1109.7$	$y_{1max} = 0.00019\ m$
$u_2 = -8.3281\ 10^{-5}$		Barra 2
$v_2 = 1\ 10^{-3}$		$M_{2max} = 2778\ Ncm$
$\Theta_2 = 0.000667$		$y_{2max} = 0.0005\ m$
$u_3 = 8.3281\ 10^{-5}$		Barra 3
$v_3 = 1\ 10^{-3}$		$M_{3max} = 2221\ Ncm$
$\Theta_3 = -0.000667$		$y_{3max} = 0.00019\ m$
$u_4 = 0$	$H_4 = 16.7$	
$v_4 = 0$	$V_4 = -100$	

Tabla comparativa de desplazamientos:

Ed-Tridim(m)	UCMMv2.0(cm)
$u_1 = 0$	$u_1 = 0$
$v_1 = 0$	$v_1 = 0$
$\Theta_1 = 0$	$\Theta_1 = 0$
$u_2 = -8.33\ 10^{-7}$	$u_2 = -8.3281\ 10^{-5}$
$v_2 = 1\ 10^{-5}$	$v_2 = 1\ 10^{-3}$
$\Theta_2 = 0.000667$	$\Theta_2 = 0.000667$
$u_3 = 8.33\ 10^{-7}$	$u_3 = 8.3281\ 10^{-5}$
$v_3 = 1\ 10^{-5}$	$v_3 = 1\ 10^{-3}$
$\Theta_3 = -0.000667$	$\Theta_3 = -0.000667$
$u_4 = 0$	$u_4 = 0$
$v_4 = 0$	$v_4 = 0$
$\Theta_4 = 0$	$u_1 = 0$

Tabla comparativa de reacciones:

Ed-Tridim(N)-(Nm)	UCMMv2.0(N)-(Ncm)
$H_1 = -16.7$	$H_1 = -16.7$
$V_1 = -100$	$V_1 = -100$
$M_1 = 11.1$	$M_1 = 1109.7$
$H_4 = 16.7$	$H_4 = 16.7$
$V_4 = 100$	$V_4 = -100$
$M_4 = 11.1$	$H_1 = -16.7$

Tabla comparativa de valores máximos:

Ed-Tridim	UCMMv2.0
Barra 1	Barra 1
$M_{1max} = 22.2\ Nm$	$M_{1max} = 2221\ Ncm$
$y_{1max} = 0.00019\ m$	$y_{1max} = 0.00019\ m$
Barra 2	Barra 2
$M_{2max} = 27.8\ Nm$	$M_{2max} = 2778\ Ncm$
$y_{2max} = 0.0005\ m$	$y_{2max} = 0.0005\ m$
Barra 3	Barra 3
$M_{3max} = 22.2\ Nm$	$M_{3max} = 2221\ Ncm$
$y_{3max} = 0.00019\ m$	$y_{3max} = 0.00019\ m$

Comprobamos que el funcionamiento para la primera estructura reticulada es satisfactorio, obteniendo resultados en reacciones y desplazamientos en algunos casos idénticos.

5.5. Estructura reticulada nº2

En este caso, añadiremos una barra a la estructura anterior y comprobaremos para una fuerza triangularmente distribuida.

Longitud barras (cm)	Área (cm2)	Módulo Elástico(N/cm2)	Momento de Inercia(cm4)	Carga (N)
200	100	200.000	833.33	100

5.5.1. Resolución con Ed-Tridim

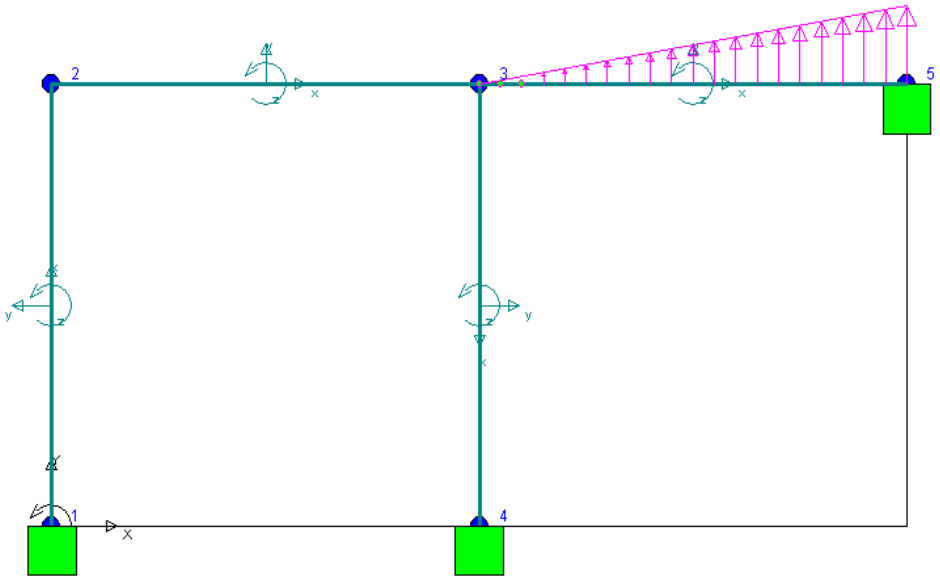


Figura 95. Estructura 5.

Momento Flector:

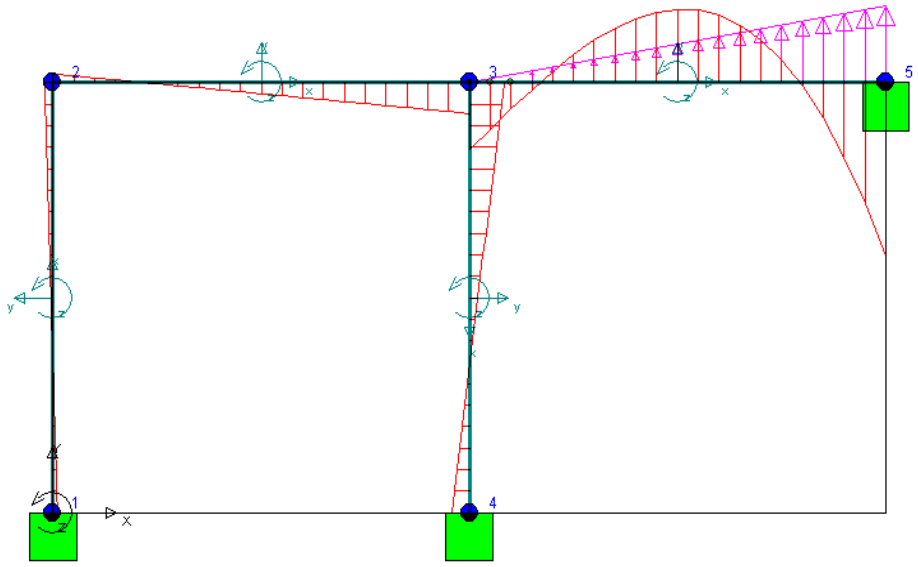


Figura 96. Diagrama de momentos estructura5 por Ed-Tridim.

Tabla de desplazamientos, reacciones y valores máximos:

Desplazamientos(m)(rad)	Reacciones (N)(Nm)	Momentos y Flechas
$u_1 = 0$	$H_1 = \ 0.842$	Barra 1
$v_1 = 0$	$V_1 = 2.56$	$M_{1max} = 1.12\ Nm$
$\Theta_1 = \ 0$	$M_1 = -0.563$	$y_{1max} = 1.1\ 10^{-5}\ m$
$u_2 = -1.79\ 10^{-7}$		Barra 2
$v_2 = -2.56\ 10^{-7}$		$M_{2max} = 4\ Nm$
$\Theta_2 = \ -3.35\ 10^{-5}$		$y_{2max} = 4.4\ 10^{-5}\ m$
$u_3 = -2.63\ 10^{-7}$		Barra 3
$v_3 = 2.9\ 10^{-6}$		$M_{3max} = 4.63Nm$
$\Theta_3 = \ 0.00139$		$y_{3max} = 4.11\ 10^{-5}\ m$
$u_4 = 0$	$H_4 = -3.47$	Barra 4
$v_4 = 0$	$V_4 = -29$	$M_{4max} = 22.4\ Nm$
$\Theta_4 = \ 0$	$M_4 = 2.31$	$y_{4max} = 0.000161\ m$
$u_5 = 0$	$H_5 = 2.63$	
$v_5 = 0$	$V_5 = -73.5$	
$\Theta_5 = \ 0$	$M_5 = 22.4$	

5.5.2. Resolución con UCMMv2.0

Momento Flector:

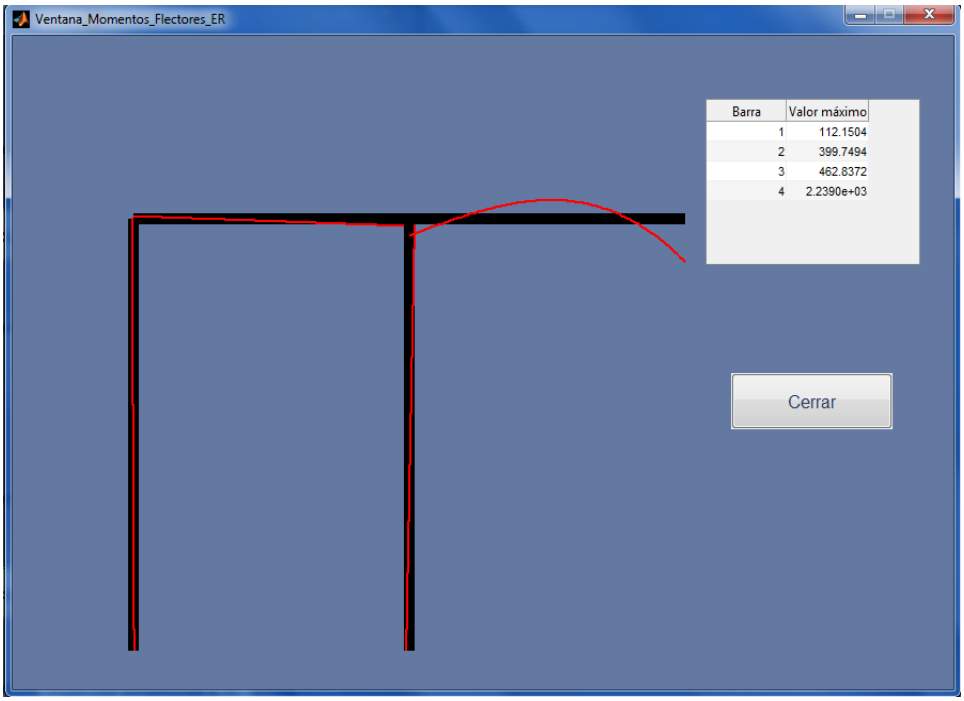


Figura 97.Diagrama de momentos estructura5 por UCMMv2.0.

Tabla de desplazamientos, reacciones y valores máximos:

Desplazamientos(cm)(rad)	Reacciones (N)(Ncm)	Momentos y Flechas
$u_1 = 0$	$H_1 = \ 0.8422$	Barra 1
$v_1 = 0$	$V_1 = 2.5595$	$M_{1max} = 112\ Ncm$
$\Theta_1 = \ 0$	$M_1 = -56.29$	$y_{1max} = 1.1\ 10^{-5}\ m$
$u_2 = -1.7851\ 10^{-5}$		Barra 2
$v_2 = -2.5595\ 10^{-5}$		$M_{2max} = 399.9\ Ncm$
$\Theta_2 = \ -3.3511\ 10^{-5}$		$y_{2max} = 4.4\ 10^{-5}\ m$
$u_3 = -2.627\ 10^{-5}$		Barra 3
$v_3 = 2.9011\ 10^{-4}$		$M_{3max} = 462.8\ Ncm$
$\Theta_3 = \ 0.000139$		$y_{3max} = 4.11\ 10^{-5}\ m$
$u_4 = 0$	$H_4 = -3.4696$	Barra 4
$v_4 = 0$	$V_4 = -29.01$	$M_{4max} = 2239\ Ncm$
$\Theta_4 = \ 0$	$M_4 = 231$	$y_{4max} = 0.000161\ m$
$u_5 = 0$	$H_5 = 2.63$	
$v_5 = 0$	$V_5 = -73.512$	
$\Theta_5 = \ 0$	$M_5 = 2240$	

Tabla comparativa de desplazamientos:

Ed-Tridim (m)(rad)	UCMMv2.0(cm)(rad)
$u_1 = 0$	$u_1 = 0$
$v_1 = 0$	$v_1 = 0$
$\Theta_1 = \ 0$	$\Theta_1 = \ 0$
$u_2 = -1.79\ 10^{-7}$	$u_2 = -1.7851\ 10^{-5}$
$v_2 = -2.56\ 10^{-7}$	$v_2 = -2.5595\ 10^{-5}$
$\Theta_2 = \ -3.35\ 10^{-5}$	$\Theta_2 = \ -3.3511\ 10^{-5}$
$u_3 = -2.63\ 10^{-7}$	$u_3 = -2.627\ 10^{-5}$
$v_3 = 2.9\ 10^{-6}$	$v_3 = 2.9011\ 10^{-4}$
$\Theta_3 = \ 0.00139$	$\Theta_3 = \ 0.000139$
$u_4 = 0$	$u_4 = 0$
$v_4 = 0$	$v_4 = 0$
$\Theta_4 = \ 0$	$\Theta_4 = \ 0$
$u_5 = 0$	$u_5 = 0$
$v_5 = 0$	$v_5 = 0$
$\Theta_5 = \ 0$	$\Theta_5 = \ 0$

Tabla de reacciones:

Ed-Tridim (m)(rad)	UCMMv2.0(cm)(rad)
$H_1 = \quad 0.842$	$H_1 = \quad 0.8422$
$V_1 = 2.56$	$V_1 = 2.5595$
$M_1 = -0.563$	$M_1 = -56.29$
$H_4 = -3.47$	$H_4 = -3.4696$
$V_4 = -29$	$V_4 = -29.01$
$M_4 = 2.31$	$M_4 = 231$
$H_5 = 2.63$	$H_5 = 2.63$
$V_5 = -73.5$	$V_5 = -73.512$
$M_5 = 22.4$	$\mathrm{M}_5 = 2240$

Tabla de valores máximos:

Ed-Tridim	UCMMv2.0
Barra 1	Barra 1
$\mathrm{M}_{1\mathrm{max}} = 1.12$ Nm	$\mathrm{M}_{1\mathrm{max}} = 112$ Ncm
$\mathrm{y}_{1\mathrm{max}} = 1.1\ 10^{-5}$ m	$\mathrm{y}_{1\mathrm{max}} = 1.1\ 10^{-5}$ m
Barra 2	Barra 2
$\mathrm{M}_{2\mathrm{max}} = 4$ Nm	$\mathrm{M}_{2\mathrm{max}} = 399.9$ Ncm
$\mathrm{y}_{2\mathrm{max}} = 4.4\ 10^{-5}$ m	$\mathrm{y}_{2\mathrm{max}} = 4.4\ 10^{-5}$ m
Barra 3	Barra 3
$\mathrm{M}_{3\mathrm{max}} = 4.63$Nm	$\mathrm{M}_{3\mathrm{max}} = 462.8$ Ncm
$\mathrm{y}_{3\mathrm{max}} = 4.11\ 10^{-5}$ m	$\mathrm{y}_{3\mathrm{max}} = 4.11\ 10^{-5}$ m
Barra 4	Barra 4
$\mathrm{M}_{4\mathrm{max}} = 22.4$ Nm	$\mathrm{M}_{4\mathrm{max}} = 2239$ Ncm
$\mathrm{y}_{4\mathrm{max}} = 0.000161$ m	$\mathrm{y}_{4\mathrm{max}} = 0.000161$ m

Mediante el análisis de sendas estructuras reticuladas, comprobamos de nuevo que los resultados son satisfactorios, por lo que podemos declarar el programa como validado completamente, tanto para estructuras reticuladas como para estructuras articuladas.

6. CONCLUSIONES

En este proyecto, se han llevado a cabo una serie de mejoras al programa UCMM. Este programa es una interfaz gráfica realizada en código MatLab con la que se pueden resolver estructuras bidimensionales con las siguientes premisas:

- Cálculo dinámico simplificado.

-Utilización fácil e intuitiva del interfaz de usuario. Exponiendo el menú principal y demás ventanas de la interfaz de forma esquemática, de forma que el usuario pueda, en primer lugar definir el problema, y comprobar los resultados obtenidos posteriormente.

-Posibilidad de introducción de elementos disipativos en la estructura.

-Visualización de resultados de una forma sencilla, comprensible y didáctica, mediante gráficas en las que se pueden visualizar y comparar distintos valores, y tablas en las que se explica cuál es valor obtenido.

-Optimización de los cálculos.

-Minimizar las líneas de programación para conseguir un menor tiempo de resolución y visualización de resultados.

-No redundancia de introducción de datos.

-Cálculo y representación en tablas de frecuencias y modos propios de las estructuras que se planteen, lo que permite una mayor comprensión por parte del usuario del efecto de la resonancia.

-Posibilidad de modificación de la estructura.

Las mejoras introducidas, proporcionan al programa un mayor número de posibilidades a la hora de establecer condiciones de contorno a las estructuras así como una mayor precisión en la obtención de resultados. Esta nueva versión del programa será denominada UCMMv2.0.

Todas las mejoras serán llevadas a cabo en estructuras articuladas y reticuladas.

La interfaz creada, mantendrá el mismo formato del programa inicial.

Todas las mejoras, fueron igualmente mejoradas mediante cálculo analítico y mediante otro programa informático de la misma índole llamado Ed-Tridim.

Estas mejoras las podremos catalogar en dos tipos diferentes:

- Mejoras en la inserción de datos:
    - Inserción de cargas distribuidas.
    - Inserción de cargas triangulares.
    - Inserción de cargas puntuales en barras.

- Mejoras en la visualización de datos:
    - Representación de la ley de momentos flectores.
    - Representación de la deformada.

7. TRABAJOS FUTUROS

Este proyecto será la primera mejora de un proyecto de mayor amplitud, que tendrá como finalidad la realización de los siguientes objetivos:

- Cálculo de tensiones en barras de la estructura.
- Resolución de estructuras en tres dimensiones.
- La posibilidad de añadir cargas distribuidas con diferentes formas.
- Posibilidad de introducción de elementos con comportamiento viscoelástico y elastoplástico.
- Posibilidad de introducción de distintos tipos de fuerzas dinámicas aplicadas.
- Validación del programa en condiciones dinámicas.

Además, se pueden realizar mejoras en la interfaz:

- Convertir UCMM en un programa ejecutable *.exe, que permita su utilización sin necesidad de tener instalado el MatLab.
- Mejora de la programación de UCMMv2.0 con un lenguaje orientado a objetos.
- Posibilidad de guardar e imprimir resultados obtenidos.
- Posibilidad de ejecutar este programa directamente desde internet.

8. BIBLIOGRAFÍA

[1] Javier García de Jalón (1997) "Aprenda MatLab 4.2 como si estuviera en Primero". Escuela Superior de ingenieros Industriales de Navarra.

[2]Prontuario Ensidesa. Manual para cálculo de estructuras metálica. Tomo I.

[3] Daniel de León Cardenas, Jorge Espinosa Caballero, José Luis Vargas Cruz. (2002) "Interfaces Gráficas en Matlab usando GUIDE". Universidad Autónoma de Baja California. Unidad Tijuana.

[4] Verónica Veas B. "Deformación en vigas". Universidad de Chile.

[5] David Fernández. "Práctica Nº1: Estructuras Articuladas". Universidad Carlos III de Madrid. Departamento de Mecánica de Medios Continuos y Teoría de Estructuras.

[6]http://www.uhu.es/josemiguel.davila/TeoriaEstructuras_archivos/TeoriaEstructuras_TEMAIII-06_DiagramasElementales.pdf. José Miguel Dávila. Universidad de Huelva.

[7] http://matpic.com/esp/matlab/matlab.html . Manual de interfaz gráfica de usuario (GUIDE) en Matlab. Parte I.

9. ANEXOS

Anexo 1

En este primer anexo del documento, se adjuntarán las funciones que fueron creadas para la descomposición de las cargas en fuerzas nodales y para la representación de los esfuerzos.

Cálculo de fuerzas nodales en barras articuladas.

```
function [Fd]=Calculo_Reacciones_Barras_EA(n,b,Pd,Pt,Pp)
global Rd Rt Rp R
numeronodos=size(n);
N=size(b);
for i=1:N(1,1)
  barra(i,:)=[n((b(i,1)),:) n((b(i,2)),:)];
end
%  Datos  de cada barra
for i=1:N(1,1)
   L(i,1)= (((barra(i,3)-barra(i,1))^2) + ((barra(i,4)-barra(i,2))^2))^0.5;
   angulo(i,1) = atan((barra(i,4)-barra(i,2))/(barra(i,3)-barra(i,1)));
end
%   Ajuste ángulo alfa de cada barra
for i=1:N(1,1)
    if ((barra(i,1))<(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
       alfa(i,:)=0;
   elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
      alfa(i,:)=angulo(i,:);
   elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
       alfa(i,:)=pi/2;
   elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
       alfa(i,:)=(pi/2)+angulo(i,:);
   elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
        alfa(i,:)=pi;
   elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
         alfa(i,:)=pi+angulo(i,:);
   elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
      alfa(i,:)=(3*pi)/2;
   elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))>(barra(i,4)))

      alfa(i,:)=(2*pi)+angulo(i,:);
    end
end
for i=1:N(1,1)
alfaR(i)=alfa(i)+ pi/2;
end
%      Vector de reacciones
for i = 1:N(1,1)
   R1d=Pd(i)*L(i)/2;
   Rd(i,:)=[ R1d*cos(alfaR(i)) R1d*(sin(alfaR(i)))  R1d*cos(alfaR(i)) R1d*(sin(alfaR(i)))];
```

```
 end
for i=1:N(1,1)
   R1t=Pt(i)*L(i)/6;
   R2t=Pt(i)*L(i)/3;
   Rt(i,:)=[ R1t*cos(alfaR(i)) R1t*(sin(alfaR(i)))  R2t*cos(alfaR(i)) R2t*(sin(alfaR(i)))];
end
for i = 1:N(1,1)
   R1p=1*(Pp(i,1)-(Pp(i,1)*Pp(i,2)));
   R2p=1*(Pp(i,1)*Pp(i,2));
   Rp(i,:) = [ R1p*cos(alfaR(i)) R1p*(sin(alfaR(i))) R2p*cos(alfaR(i))  R2p*(sin(alfaR(i)))];
end
R=Rd+Rt+Rp;
Fd=zeros(numeronodos(1,1),2);
for i=1:1:numeronodos(1,1)
   %Horizontales
   for j=1:1:N(1,1)
      if b(j,1)==i
         Fd(i,1)=Fd(i,1)+R(j,1);
      end
      if b(j,2)==i
         Fd(i,1)=Fd(i,1)+R(j,3);
      end
   end
   %Verticales
    for j=1:N(1,1)
      if b(j,1)==i
         Fd(i,2)=Fd(i,2)+R(j,2);
      end
      if b(j,2)==i
         Fd(i,2)=Fd(i,2)+R(j,4);
      end
    end
end
disp(Fd);
end
```

Cálculo de fuerzas nodales en estructuras reticuladas

```
function [FdR]=Calculo_Reacciones_ER(n,b,Pd,Pt,Pp)
%   Datos iniciales, matriz de nodos
% n = [0 0; 1 1;2 0];
% numeronodos=size(n);
%   Coordenadas en nodos de cada barra
% b = [1 2;2 3];
%   Matrices de cargas, en el caso de cargas puntuales, es necesario
%   insertar la distancia a la que se encuentra la carga, desde el nodo
%   inicial y en función de L
```

```
%Pd=[100;0];
%Pt=[0;100];
%Pp=[0 0;0 0];
global N numeronodos
numeronodos=size(n);
N=size(b);
for i=1:N(1,1)
  barra(i,:)=[n((b(i,1)),:) n((b(i,2)),:)];
end
%  Datos  de cada barra
for i=1:N(1,1)
   L(i,1)= (((barra(i,3)-barra(i,1))^2) + ((barra(i,4)-barra(i,2))^2))^0.5;
   angulo(i,1) = atan((barra(i,4)-barra(i,2))/(barra(i,3)-barra(i,1)));
end
%   Ajuste ángulo alfa de cada barra
for i=1:N(1,1)
    if ((barra(i,1))<(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
       alfa(i,:)=0;
   elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
      alfa(i,:)=angulo(i,:);
   elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
       alfa(i,:)=pi/2;
   elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
       alfa(i,:)=(pi/2)+angulo(i,:);
   elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
        alfa(i,:)=pi;
   elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
         alfa(i,:)=pi+angulo(i,:);
   elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
      alfa(i,:)=(3*pi)/2;
   elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))>(barra(i,4)))

      alfa(i,:)=(2*pi)+angulo(i,:);
    end
end
for i=1:N(1,1)
alfaR(i)=alfa(i)+ pi/2;
end
%      Vector de reacciones
for i = 1:N(1,1)
   R1d=Pd(i)*L(i)/2;
   Gd=(Pd(i)*L(i)^2)/12;
   Rd(i,:)=[ R1d*cos(alfaR(i)) R1d*(sin(alfaR(i))) Gd R1d*cos(alfaR(i)) R1d*sin(alfaR(i)) -
Gd ];
 end
for i=1:N(1,1)
   R1t=3*Pt(i)*L(i)/20;
   R2t=7*Pt(i)*L(i)/20;
```

```
   G1t=(Pt(i)*L(i)^2)/30;
   G2t=(Pt(i)*L(i)^2)/20;
   Rt(i,:)=[ R1t*cos(alfaR(i)) R1t*(sin(alfaR(i))) G1t  R2t*cos(alfaR(i)) R2t*(sin(alfaR(i))) -
G2t ];
end
for i = 1:N(1,1)
   R1p=1*((Pp(i,1)*(L(i)-(L(i)*Pp(i,2)))^2)/L(i)^3)*(L(i)+(2*Pp(i,2)*L(i)));
   R2p=1*((Pp(i,1)*(L(i)*Pp(i,2))^2)/L(i)^3)*(L(i)+(2*(L(i)-(L(i)*Pp(i,2)))));
   G1p=(Pp(i,1)*(L(i)*Pp(i,2))*(L(i)-(L(i)*Pp(i,2)))^2)/L(i)^2;
   G2p=(Pp(i,1)*((L(i)*Pp(i,2))^2)*(L(i)-(L(i)*Pp(i,2))))/L(i)^2;
   Rp(i,:) = [ R1p*cos(alfaR(i)) R1p*(sin(alfaR(i))) G1p R2p*cos(alfaR(i))
R2p*(sin(alfaR(i))) -G2p];
end
R=Rd+Rt+Rp;

FdR=zeros(numeronodos(1,1),3);
for i=1:1:numeronodos(1,1)
   %Horizontales
   for j=1:1:N(1,1)
      if b(j,1)==i
         FdR(i,1)=FdR(i,1)+R(j,1);
      end
      if b(j,2)==i
         FdR(i,1)=FdR(i,1)+R(j,4);
      end
   end
   %Verticales
    for j=1:N(1,1)
      if b(j,1)==i
         FdR(i,2)=FdR(i,2)+R(j,2);
      end
      if b(j,2)==i
         FdR(i,2)=FdR(i,2)+R(j,5);
      end
    end
    % Momentos
     for j=1:N(1,1)
      if b(j,1)==i
         FdR(i,3)=FdR(i,3)+R(j,3);
      end
      if b(j,2)==i
         FdR(i,3)=FdR(i,3)+R(j,6);
      end
    end
end
disp(FdR);
end
```

Representación de momentos flectores en estructuras articuladas.

```
function [M]=Calculo_Momentos_Flectores_EA(n,b,Pd,Pt,Pp)
%   Datos iniciales, matriz de nodos
% n = [0 0; 1 1;2 0];
% numeronodos=size(n);
%   Coordenadas en nodos de cada barra
% b = [1 2;2 3];
%   Matrices de cargas, en el caso de cargas puntuales, es necesario
%   insertar la distancia a la que se encuentra la carga, desde el nodo
%   inicial y en función de L
%Pd=[0;0];
%Pt=[0;0];
%Pp=[0 0;-100 0.5];
%   Matriz de coordenadas iniciales y finales de cada barra
global N
numeronodos=size(n);
N=size(b);
for i=1:N(1,1)
  barra(i,:)=[n((b(i,1)),:) n((b(i,2)),:)];
end
%  Datos  de cada barra
for i=1:N(1,1)
   L(i,1)= (((barra(i,3)-barra(i,1))^2) + ((barra(i,4)-barra(i,2))^2))^0.5;
   angulo(i,1) = atan((barra(i,4)-barra(i,2))/(barra(i,3)-barra(i,1)));
end
%   Ajuste ángulo alfa de cada barra
for i=1:N(1,1)
    if ((barra(i,1))<(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
       alfa(i,:)=0;
   elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
      alfa(i,:)=angulo(i,:);
   elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
       alfa(i,:)=pi/2;
   elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
       alfa(i,:)=(pi)+angulo(i,:);
   elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
        alfa(i,:)=pi;
   elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
         alfa(i,:)=pi+angulo(i,:);
   elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
      alfa(i,:)=(3*pi)/2;
   elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))>(barra(i,4)))

      alfa(i,:)=(2*pi)+angulo(i,:);
    end
end
%   Particiones barra
for i =1 : N(1,1)
```

```
   Li(i,:) = L(i,1) /100;
   xL(i,:)=(0:Li(i,:):L(i));
end
%   barras
for i=1:N(1,1)
   for j=1 : 101
   xb(i,j)= barra(i,1) + xL(i,j) * cos(alfa(i,1));
   yb(i,j)= barra(i,2) + xL(i,j) * sin(alfa(i,1));
   end
end
%   Momento en barras
for i = 1:N(1,1)
   for j=1:101
      Md(i,j)= (((Pd(i)*L(i))/2)*xL(i,j))- (Pd(i)*xL(i,j)^2)/2;
      Mt(i,j)=(-1*(Pt(i)*(xL(i,j)^3))/(6*L(i))) + ((Pt(i)*xL(i,j)*L(i))/(6));
   end

end
% Momento en barras con una carga puntual
for i =1:N(1,1)
   for j=1:101
       if (xL(i,j)/L(i))< Pp(i,2)
          Mp(i,j)=( (Pp(i,1)*(L(i)-(L(i)*Pp(i,2)))*xL(i,j))/L(i));
       else
          Mp(i,j)=(-1*( Pp(i,1)*(xL(i,j)- L(i)*Pp(i,2))) + ((Pp(i,1)*(L(i)-
(L(i)*Pp(i,2)))*xL(i,j))/L(i)));
       end
   end
end
global M
M=Mt+Md+Mp;
 %  Factor de escala
 E=max(max(abs(M)))/L(1);
 %  Escalado
for i=1:N(1,1)
   ME(i,:)=M(i,:)/(E*10);
end
%  Giro momentos
for i=1:N(1,1)
   for j=1:101
   radio(i,j)=sqrt((xL(i,j)^2)+(ME(i,j)^2));
   theta(i,j)= atan(ME(i,j)/xL(i,j)) + alfa(i);
   xm(i,j)= barra(i,1)+ radio(i,j)*cos(theta(i,j));
   ym(i,j)= barra(i,2) + radio(i,j)*sin(theta(i,j));
   end
end
hold on
for i=1:N(1,1)
```

```
    plot(xb(i,:),yb(i,:),'k','linewidth',8)
    plot(xm(i,:),ym(i,:),'r','linewidth',2)
end
axis off
end
```

Representación de Momentos flectores en reticuladas.

```
function
[MR]=Calculo_Leyes_Momento_Flector_ER(n,b,RestriccionesNodo,Pd,Pt,Pp,Desp,E,A,I)
%Datos necesarios:

  % Matriz de nodos
  %   n = [0 0; 100 100; 200 0];
  % Matriz de barras
  %   b=[1 2; 2 3];
  %Restricciones en los nodos
  %   RestriccionesNodo=[0 0 0;1 1 1;0 0 0];
  %Cargas aplicadas:
  %   Pd=[1;0];
  %   Pt=[0;0];
  %   Pp=[0 0;0 0];
  %Desplazamientos producidos en los movimientos no restringidos de los nodos
  %   Desp=[2.6483e-04;-3.5179e-04  ; 1.7479e-04];nme

N=size(b);
numeronodos=size(n);
for i=1:N(1,1)
  barra(i,:)=[n((b(i,1)),:) n((b(i,2)),:)];
end
%  Datos  de cada barra
for i=1:N(1,1)
   L(i,1)= (((barra(i,3)-barra(i,1))^2) + ((barra(i,4)-barra(i,2))^2))^0.5;
   angulo(i,1) = atan((barra(i,4)-barra(i,2))/(barra(i,3)-barra(i,1)));
end
%   Ajuste ángulo alfa de cada barra
for i=1:N(1,1)
    if ((barra(i,1))<(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
       alfa(i,:)=0;
   elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
      alfa(i,:)=angulo(i,:);
   elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
       alfa(i,:)=pi/2;
   elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
       alfa(i,:)=(pi/2)+angulo(i,:);
   elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
        alfa(i,:)=pi;
   elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
         alfa(i,:)=pi+angulo(i,:);
   elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
```

```
      alfa(i,:)=(3*pi)/2;
   elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))>(barra(i,4)))

      alfa(i,:)=(2*pi)+angulo(i,:);
    end
end
%   Particiones barra
for i =1 : N(1,1)
   Li(i,:) = L(i,1) /100;
   xL(i,:)=(0:Li(i,:):L(i));
end
%   barras para el dibujo
for i=1:N(1,1)
   for j=1 : 101
   xb(i,j)= barra(i,1) + xL(i,j) * cos(alfa(i,1));
   yb(i,j)= barra(i,2) + xL(i,j) * sin(alfa(i,1));
   end
end


%Vector desplazamiento completo organizado en filas, cada fila tiene 3
%columnas, correspondientes a los 3 tipos de desplazamientos que tiene cada
%nodo
global Desplazamientosentero
Desplazamientosentero=zeros(numeronodos(1,1),3);
contadordesp=1;
Restriccionesnododesp=RestriccionesNodo;
for i =1:numeronodos
   for j=1:3;
    if Restriccionesnododesp(i,j)==0
      Desplazamientosentero(i,j)=0;

    end
    if Restriccionesnododesp(i)==1
      Desplazamientosentero(i,j)=Desplazamientosentero(i,j)+Desp(contadordesp,1);
      contadordesp=1+contadordesp;

    end
   end
end


%Vector de esfuerzos de cada barra, que salen de la multiplicación de su
%matriz de rigidez y los desplazamientos de los nodos inicial y final
global FdptR
for i =1:N(1,1)
k=kglobalesreticuladasb(E(i),A(i),L(i),I(i),alfa(i));
```

```
u(i,:)= [Desplazamientosentero(b(i,1),:) Desplazamientosentero(b(i,2),:) ];
U=u';


F(:,i)=k*U(:,i);

RNODO(i,:)=[RestriccionesNodo(b(i,1),:) RestriccionesNodo(b(i,2),:)];
RN=RNODO';
FNODO(i,:)=[FdptR(b(i,1),:) FdptR(b(i,2),:)];
FN=FNODO';
end

F1=F;

%   sumatorio del vector de fuerzas y reacciones.
for i=1:6
   for j=1:N(1,1)
      if RN(i,j)==0
         F1(i,j)=F(i,j)-FN(i,j);
      elseif RN(i,j)==1
        F1(i,j)=F(i,j)-FN(i,j);
      end
   end
end
% Giro Matriz F

for i=1:N(1,1)
   Fg(i,:) = [ (F(1,i)*cos(alfa(i))+F(2,i)*sin(alfa(i))) (-F(1,i)*sin(alfa(i))+F(2,i)*cos(alfa(i)))
F(3,i) (F(4,i)*cos(alfa(i))+F(5,i)*sin(alfa(i))) (-F(4,i)*sin(alfa(i))+F(5,i)*cos(alfa(i))) F(6,i)];
end

for i=1:N(1,1)
   Fg1(i,:) = [ (F1(1,i)*cos(alfa(i))+F1(2,i)*sin(alfa(i))) (-
F1(1,i)*sin(alfa(i))+F1(2,i)*cos(alfa(i))) F1(3,i) (F1(4,i)*cos(alfa(i))+F1(5,i)*sin(alfa(i))) (-
F1(4,i)*sin(alfa(i))+F1(5,i)*cos(alfa(i))) F1(6,i)];
end


% Calculo del momento producido por una carga distribuida
for i = 1:N(1,1)
   for j=1:101
      if Pd(:,:)==0
            Md(i,j)=0;
      else if Pd(i)==0
            Md(i,j)= -((Pd(i)*(xL(i,j)^2))/2)-(Fg(i,2)*xL(i,j)) +Fg(i,3)  ;
         else if Pd(i)~=0
            Md(i,j)= -((Pd(i)*(xL(i,j)^2))/2)-(Fg1(i,2)*xL(i,j)) +Fg1(i,3)  ;
            end
```

```
        end
      end

  end
end

% Calculo del momento producido por una carga triangular
for i=1:N(1,1)
  for j=1:101
    if Pt(:,:)==0
      Mt(i,j)=0;
    else if Pt(i)~=0
       Mt(i,j)= -(Pt(i)*xL(i,j)^3)/(6*L(i))-(Fg1(i,2)*xL(i,j))+ Fg1(i,3);
      else if Pt(i)==0
      Mt(i,j)= -(Pt(i)*xL(i,j)^3)/(6*L(i))-(Fg(i,2)*xL(i,j))+ Fg(i,3);
         end
      end
    end
  end
end

% Momento en barras con una carga puntual
for i =1:N(1,1)
  for j=1:101
     if Pp(i,1)==0
       if (xL(i,j)/L(i))< Pp(i,2)
        Mp(i,j)= Fg(i,3) -(Fg(i,2)*xL(i,j));
       else
        Mp(i,j)= -(Pp(i,1)*(xL(i,j)-Pp(i,2)*L(i))) + Fg(i,3)- (Fg(i,2)*xL(i,j));
       end
      else if Pp(i,1)~=0
       if (xL(i,j)/L(i))< Pp(i,2)
        Mp(i,j)= Fg1(i,3) -(Fg1(i,2)*xL(i,j));
       else
        Mp(i,j)= -(Pp(i,1)*(xL(i,j)-Pp(i,2)*L(i))) + Fg1(i,3)- (Fg1(i,2)*xL(i,j));
       end
      end
    end
  end
end

for i=1:N(1,1)
  for j=1:101
    if Pp(:,:)==0
       Mp(i,j)=0;
    end
  end
end
```

```
global M
M=Md+Mt+Mp;

 %  Factor de escala
 Esc=(L(1)/10)/max(max(abs(M)));
 %  Escalado
for i=1:N(1,1)
   ME(i,:)=M(i,:)*Esc;

end
%  Giro momentos para su representación
for i=1:N(1,1)
   for j=1:101
   radio(i,j)=sqrt((xL(i,j)^2)+(ME(i,j)^2));
   theta(i,j)= atan(ME(i,j)/xL(i,j)) + alfa(i);
   xm(i,j)= barra(i,1)+ radio(i,j)*cos(theta(i,j));
   ym(i,j)= barra(i,2) + radio(i,j)*sin(theta(i,j));
   end
end
%Representación figura
hold on
for i=1:N(1,1)
     plot(xb(i,:),yb(i,:),'k','linewidth',8)
     plot(xm(i,:),ym(i,:),'r','linewidth',2)
end
  axis off
end
```

Representación flecha en articuladas.

```
function
[V]=Calculo_Flecha_EA(n,b,Pd,Pt,Pp,ReestriccionesNodo,DesplazamientoXarticulada,E,
I)
%   Datos iniciales, matriz de nodos
% n = [0 0; 1 1;2 0];
% numeronodos=size(n);
%ReestriccionesNodo=[0 0 ; 1 1 ; 0 0];
%DesplazamientoXarticulada=[ -0.0354 ; -0.0354 ];
%   Coordenadas en nodos de cada barra
 %  b = [1 2;2 3];
 %  Matriz de Modulo elástico
% E=[200000 ; 200000];
 % I=[833;833];
%   Matrices de cargas, en el caso de cargas puntuales, es necesario
%   insertar la distancia a la que se encuentra la carga, desde el nodo
%   inicial y en función de L
```

```
% Pd=[0;0];
% Pt=[0;0];
%Pp=[100 0.25;0 0];
%   Matriz de coordenadas iniciales y finales de cada barra
global N numeronodos
numeronodos=size(n);
N=size(b);
nd=n;



for i=1:N(1,1)
  barra(i,:)=[n((b(i,1)),:) n((b(i,2)),:)];
end
%  Datos  de cada barra
for i=1:N(1,1)
   L(i,1)= (((barra(i,3)-barra(i,1))^2) + ((barra(i,4)-barra(i,2))^2))^0.5;
   angulo(i,1) = atan((barra(i,4)-barra(i,2))/(barra(i,3)-barra(i,1)));
end
%   Ajuste ángulo alfa de cada barra
for i=1:N(1,1)
    if ((barra(i,1))<(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
       alfa(i,:)=0;
   elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
      alfa(i,:)=angulo(i,:);
   elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
       alfa(i,:)=pi/2;
   elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
       alfa(i,:)=pi+angulo(i,:);
   elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
        alfa(i,:)=pi;
   elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
         alfa(i,:)=pi+angulo(i,:);
   elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
      alfa(i,:)=(3*pi)/2;
   elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))>(barra(i,4)))

      alfa(i,:)=(2*pi)+angulo(i,:);
    end
end
%   Particiones barra
for i =1 : N(1,1)
   Li(i,:) = L(i,1) /100;
   xL(i,:)=(0:Li(i,:):L(i));
end

%   barras
for i=1:N(1,1)
```

```
   for j=1 : 101
   xb(i,j)= barra(i,1) + xL(i,j) * cos(alfa(i,1));
   yb(i,j)= barra(i,2) + xL(i,j) * sin(alfa(i,1));
   end
end
 %  Factor de escala  vector desplazamientos
 Esc=(L(1)/10)/max(max(abs(DesplazamientoXarticulada)));
 %  Escalado desplazamientos
DesplazamientoXarticulada(:,:)=DesplazamientoXarticulada(:,:)*Esc;


contadordef=1;
for i = 1:1:numeronodos
   for j =1:2
    if ReestriccionesNodo(i,j)==0
       nd(i,j)=n(i,j);
    else if ReestriccionesNodo(i,j) == 1
       nd(i,j)=n(i,j)+ DesplazamientoXarticulada(contadordef,1);
       contadordef=contadordef+1;
       end
    end
   end
end

for i=1:N(1,1)
  barradef(i,:)=[nd((b(i,1)),:) nd((b(i,2)),:)];
end
%  Datos  de cada barra
for i=1:N(1,1)
   Ldef(i,1)= (((barradef(i,3)-barradef(i,1))^2) + ((barradef(i,4)-barradef(i,2))^2))^0.5;
   angulodef(i,1) = atan((barradef(i,4)-barradef(i,2))/(barradef(i,3)-barradef(i,1)));
end
%   Ajuste ángulo alfa de cada barra
for i=1:N(1,1)
    if ((barradef(i,1))<(barradef(i,3))) && ((barradef(i,2))==(barradef(i,4)))
       alfadef(i,:)=0;
   elseif ((barradef(i,1))<(barradef(i,3))) && ((barradef(i,2))<(barradef(i,4)))
      alfadef(i,:)=angulodef(i,:);
   elseif ((barradef(i,1))==(barradef(i,3))) && ((barradef(i,2))<(barradef(i,4)))
       alfadef(i,:)=pi/2;
   elseif ((barradef(i,1))>(barradef(i,3))) && ((barradef(i,2))<(barradef(i,4)))
       alfadef(i,:)=pi+angulodef(i,:);
   elseif ((barradef(i,1))>(barradef(i,3))) && ((barradef(i,2))==(barradef(i,4)))
        alfadef(i,:)=pi;
   elseif ((barradef(i,1))>(barradef(i,3))) && ((barradef(i,2))>(barradef(i,4)))
         alfadef(i,:)=pi+angulodef(i,:);
   elseif ((barradef(i,1))==(barradef(i,3))) && ((barradef(i,2))>(barradef(i,4)))
      alfadef(i,:)=(3*pi)/2;
```

```
   elseif ((barradef(i,1))<(barradef(i,3))) && ((barradef(i,2))>(barradef(i,4)))

      alfadef(i,:)=(2*pi)+angulodef(i,:);
    end
end

%   Particiones barra
for i =1 : N(1,1)
   Lidef(i,:) = Ldef(i,1) /100;
   xLdef(i,:)=(0:Lidef(i,:):Ldef(i));
end

%   barras
for i=1:N(1,1)
   for j=1 : 101
   xbdef(i,j)= barradef(i,1) + xLdef(i,j) * cos(alfadef(i,1));
   ybdef(i,j)= barradef(i,2) + xLdef(i,j) * sin(alfadef(i,1));
   end
end
%   Flecha en barras por PD y PT
for i = 1:N(1,1)
   for j=1:101
      Vd(i,j)=(Pd(i)*xL(i,j)*((xL(i,j)^3)-(2*L(i)*xL(i,j)^2)+(L(i)^3)))/(24*E(i)*I(i));
      Vt(i,j)=((Pt(i)*xL(i,j)*(L(i)^3))/(360*E(i)*I(i)))*(7-
((10*(xL(i,j)^2))/(L(i)^2))+(3*(xL(i,j)^4))/(L(i)^4));
   end
end

% Flecha en barras por Pp

for i=1:N(1,1)
   for j=2:1:101
    if (xL(i,j)/L(i))< Pp(i,2)
      Vp(i,j)= (((Pp(i,1)*L(i)*(L(i)-(Pp(i,2)*L(i)))*xL(i,j))/(6*E(i)*I(i)))*(1-(((L(i)-
(Pp(i,2)*L(i)))^2)/L(i)^2)-((xL(i,j)^2)/(L(i)^2))));
   else
      Vp(i,j)=(((Pp(i,1)*L(i)*(Pp(i,2)*L(i))*(L(i)-xL(i,j)))/(6*E(i)*I(i)))*(1-
(((Pp(i,2)*L(i))^2)/(L(i)^2))-(((L(i)-xL(i,j))^2)/(L(i)^2))));
    end
   end
end
%global V Vd Vt Vp
V=Vd+Vt+Vp;

%   Flecha deformada en barras por PD y PT contando con desplazamiento de
%   los nodos
for i = 1:N(1,1)
   for j=1:101
```

```
      Vddef(i,j)=(Pd(i)*xLdef(i,j)*((xLdef(i,j)^3)-
(2*Ldef(i)*xLdef(i,j)^2)+(Ldef(i)^3)))/(24*E(i)*I(i));
      Vtdef(i,j)=((Pt(i)*xLdef(i,j)*(Ldef(i)^3))/(360*E(i)*I(i)))*(7-
((10*(xLdef(i,j)^2))/(Ldef(i)^2))+(3*(xLdef(i,j)^4))/(Ldef(i)^4));
   end
end

% Flecha en barras por Pp contando con el desplazamiento de los nodos

for i=1:N(1,1)
   for j=2:1:101
     if (xLdef(i,j)/Ldef(i))< Pp(i,2)
       Vpdef(i,j)= (((Pp(i,1)*Ldef(i)*(Ldef(i)-(Pp(i,2)*Ldef(i)))*xLdef(i,j))/(6*E(i)*I(i)))*(1-
(((Ldef(i)-(Pp(i,2)*Ldef(i)))^2)/Ldef(i)^2)-((xLdef(i,j)^2)/(Ldef(i)^2))));
   else
      Vpdef(i,j)=(((Pp(i,1)*Ldef(i)*(Pp(i,2)*Ldef(i))*(Ldef(i)-xLdef(i,j)))/(6*E(i)*I(i)))*(1-
(((Pp(i,2)*Ldef(i))^2)/(Ldef(i)^2))-(((Ldef(i)-xLdef(i,j))^2)/(Ldef(i)^2))));
     end
   end
end
global Vdef
Vdef=Vddef+Vtdef+Vpdef;

 %  Factor de escala  flecha
 Esc=(L(1)/10)/max(max(abs(Vdef)));
 %  Escalado flecha
for i=1:N(1,1)
   VE(i,:)=Vdef(i,:)*Esc;
end
%  Giro flecha escalada
for i=1:N(1,1)
   for j=1:101
   radio(i,j)=sqrt((xLdef(i,j)^2)+(VE(i,j)^2));
   theta(i,j)= atan(VE(i,j)/xLdef(i,j)) + alfadef(i);
   xv(i,j)= barradef(i,1)+ radio(i,j)*cos(theta(i,j));
   yv(i,j)= barradef(i,2) + radio(i,j)*sin(theta(i,j));
   end
end
XV=(xbdef+xv)/2;
YV=(ybdef+yv)/2;
hold on
for i=1:N(1,1)
     plot(xb(i,:),yb(i,:),'k','linewidth',8)
     plot(XV(i,:),YV(i,:),'g','linewidth',3)
  % plot(xbdef(i,:),ybdef(i,:),'g','linewidth',3)
end
axis off
end
```

Representación flecha en reticuladas.

```
function [V]=Calculo_Flecha_ER(n,b,Pd,Pt,Pp,RestriccionesNodo,Desp,E,I,A)
%   Datos iniciales, matriz de nodos
% n = [0 0; 1 1;2 0];
% numeronodos=size(n);
%ReestriccionesNodo=[0 0 ; 1 1 ; 0 0];
%DesplazamientoXarticulada=[ -0.0354 ; -0.0354 ];
%   Coordenadas en nodos de cada barra
 %  b = [1 2;2 3];
 %  Matriz de Modulo elástico
% E=[200000 ; 200000];
 % I=[833;833];
%   Matrices de cargas, en el caso de cargas puntuales, es necesario
%   insertar la distancia a la que se encuentra la carga, desde el nodo
%   inicial y en función de L
% Pd=[0;0];
% Pt=[0;0];
%Pp=[100 0.25;0 0];
%   Matriz de coordenadas iniciales y finales de cada barra
global N numeronodos
numeronodos=size(n);
N=size(b);
nd=n;



for i=1:N(1,1)
  barra(i,:)=[n((b(i,1)),:) n((b(i,2)),:)];
end
%  Datos  de cada barra
for i=1:N(1,1)
   L(i,1)= (((barra(i,3)-barra(i,1))^2) + ((barra(i,4)-barra(i,2))^2))^0.5;
   angulo(i,1) = atan((barra(i,4)-barra(i,2))/(barra(i,3)-barra(i,1)));
end
%   Ajuste ángulo alfa de cada barra
for i=1:N(1,1)
    if ((barra(i,1))<(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
       alfa(i,:)=0;
   elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
      alfa(i,:)=angulo(i,:);
   elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
       alfa(i,:)=pi/2;
   elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
       alfa(i,:)=pi+angulo(i,:);
   elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
        alfa(i,:)=pi;
   elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
```

```
        alfa(i,:)=pi+angulo(i,:);
   elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
      alfa(i,:)=(3*pi)/2;
   elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))>(barra(i,4)))

      alfa(i,:)=(2*pi)+angulo(i,:);
    end
end
%   Particiones barra
for i =1 : N(1,1)
   Li(i,:) = L(i,1) /100;
   xL(i,:)=(0:Li(i,:):L(i));
end

%   barras
for i=1:N(1,1)
   for j=1 : 101
   xb(i,j)= barra(i,1) + xL(i,j) * cos(alfa(i,1));
   yb(i,j)= barra(i,2) + xL(i,j) * sin(alfa(i,1));
   end
end

global Desplazamientosentero
%  Factor de escala  vector desplazamientos
 Esc=(L(1)/10)/max(max(abs(Desplazamientosentero)));
 %  Escalado desplazamientos
Desplazamientosentero(:,:)=Desplazamientosentero(:,:)*Esc;


for i=1:numeronodos
   for j=1:2
      nd(i,j)=n(i,j)+Desplazamientosentero(i,j);
   end
end
% Calculos inciales con la matriz de nodos nd que incluye los
% desplazamientos producidos.

for i=1:N(1,1)
  barradef(i,:)=[nd((b(i,1)),:) nd((b(i,2)),:)];
end
%  Datos  de cada barra con los desplazamientos
for i=1:N(1,1)
   Ldef(i,1)= (((barradef(i,3)-barradef(i,1))^2) + ((barradef(i,4)-barradef(i,2))^2))^0.5;
   angulodef(i,1) = atan((barradef(i,4)-barradef(i,2))/(barradef(i,3)-barradef(i,1)));
end
%   Ajuste ángulo alfa de cada barra
for i=1:N(1,1)
    if ((barradef(i,1))<(barradef(i,3))) && ((barradef(i,2))==(barradef(i,4)))
```

```
      alfadef(i,:)=0;
  elseif ((barradef(i,1))<(barradef(i,3))) && ((barradef(i,2))<(barradef(i,4)))
     alfadef(i,:)=angulodef(i,:);
  elseif ((barradef(i,1))==(barradef(i,3))) && ((barradef(i,2))<(barradef(i,4)))
      alfadef(i,:)=pi/2;
  elseif ((barradef(i,1))>(barradef(i,3))) && ((barradef(i,2))<(barradef(i,4)))
      alfadef(i,:)=pi+angulodef(i,:);
  elseif ((barradef(i,1))>(barradef(i,3))) && ((barradef(i,2))==(barradef(i,4)))
       alfadef(i,:)=pi;
  elseif ((barradef(i,1))>(barradef(i,3))) && ((barradef(i,2))>(barradef(i,4)))
        alfadef(i,:)=pi+angulodef(i,:);
  elseif ((barradef(i,1))==(barradef(i,3))) && ((barradef(i,2))>(barradef(i,4)))
     alfadef(i,:)=(3*pi)/2;
  elseif ((barradef(i,1))<(barradef(i,3))) && ((barradef(i,2))>(barradef(i,4)))

     alfadef(i,:)=(2*pi)+angulodef(i,:);
   end
end

%   Particiones barra
for i =1 : N(1,1)
  Lidef(i,:) = Ldef(i,1) /100;
  xLdef(i,:)=(0:Lidef(i,:):Ldef(i));
end

%   barras
for i=1:N(1,1)
  for j=1 : 101
  xbdef(i,j)= barradef(i,1) + xLdef(i,j) * cos(alfadef(i,1));
  ybdef(i,j)= barradef(i,2) + xLdef(i,j) * sin(alfadef(i,1));
  end
end


%Vector desplazamiento completo organizado en filas, cada fila tiene 3
%columnas, correspondientes a los 3 tipos de desplazamientos que tiene cada
%nodo

Desplazamientosentero=zeros(numeronodos(1,1),3);
contadordesp=1;
Restriccionesnododesp=RestriccionesNodo;
for i =1:numeronodos(1,1)
  for j=1:3;
   if Restriccionesnododesp(i,j)==0
     Desplazamientosentero(i,j)=0;

   end
   if Restriccionesnododesp(i)==1
```

```
      Desplazamientosentero(i,j)=Desplazamientosentero(i,j)+Desp(contadordesp,1);
      contadordesp=1+contadordesp;

    end
   end
end


%Vector de esfuerzos de cada barra, que salen de la multiplicación de su
%matriz de rigidez y los desplazamientos de los nodos inicial y final
global FdptR
for i =1:N(1,1)
k=kglobalesreticuladasb(E(i),A(i),L(i),I(i),alfa(i));

u(i,:)= [Desplazamientosentero(b(i,1),:) Desplazamientosentero(b(i,2),:) ];

U=u';


F(:,i)=k*U(:,i);

RNODO(i,:)=[RestriccionesNodo(b(i,1),:) RestriccionesNodo(b(i,2),:)];
RN=RNODO';
FNODO(i,:)=[FdptR(b(i,1),:) FdptR(b(i,2),:)];
FN=FNODO';
end
F1=F;

%   sumatorio del vector de fuerzas y reacciones.
for i=1:6
   for j=1:N(1,1)
      if RN(i,j)==0
         F1(i,j)=F(i,j)-FN(i,j);
      elseif RN(i,j)==1
        F1(i,j)=F(i,j)-FN(i,j);
      end
   end
end
% Giro Matriz F

for i=1:N(1,1)
   Fg(i,:) = [ (F(1,i)*cos(alfa(i))+F(2,i)*sin(alfa(i))) (-F(1,i)*sin(alfa(i))+F(2,i)*cos(alfa(i)))
F(3,i) (F(4,i)*cos(alfa(i))+F(5,i)*sin(alfa(i))) (-F(4,i)*sin(alfa(i))+F(5,i)*cos(alfa(i))) F(6,i)];
end

for i=1:N(1,1)
```

```
   Fg1(i,:) = [ (F1(1,i)*cos(alfa(i))+F1(2,i)*sin(alfa(i))) (-
F1(1,i)*sin(alfa(i))+F1(2,i)*cos(alfa(i))) F1(3,i) (F1(4,i)*cos(alfa(i))+F1(5,i)*sin(alfa(i))) (-
F1(4,i)*sin(alfa(i))+F1(5,i)*cos(alfa(i))) F1(6,i)];
end

% Calculo de la flecha producida por una carga distribuida
for i = 1:N(1,1)
   for j=1:101
      if Pd(:,:)==0
          Vd(i,j)=0;
      else if Pd(i)~=0
          Vd(i,j)= -(u(i,3)*xL(i,j)) + (((Pd(i)*(xL(i,j)^4))/8)- ((Fg1(i,2)*(xL(i,j)^3))/3)+
((Fg1(i,3)*(xL(i,j)^2))/2))/(E(i)*I(i))  ;
         else if Pd(i)==0
               Vd(i,j)= -(u(i,3)*xL(i,j)) +(((Pd(i)*(xL(i,j)^4))/8)- ((Fg(i,2)*(xL(i,j)^3))/3)
+((Fg(i,3)*(xL(i,j)^2))/2))/(E(i)*I(i))  ;
            end
         end
      end
   end
end
% Calculo del momento producido por una carga triangular
for i=1:N(1,1)
   for j=1:101
      if Pt(:,:)==0
        Vt(i,j)=0;
      else if Pt(i)~=0
          Vt(i,j)= (u(i,3)*xL(i,j)) +(-((Pt(i)*xL(i,j)^5)/(30*L(i)))-((Fg1(i,2)*(xL(i,j)^3))/3)+
((Fg1(i,3)*(xL(i,j)^2))/2))/(E(i)*I(i));
         else if Pt(i)==0
               Vt(i,j)= (u(i,3)*xL(i,j)) +(-((Pt(i)*xL(i,j)^5)/(30*L(i)))-((Fg(i,2)*(xL(i,j)^3))/3)+
((Fg(i,3)*(xL(i,j)^2))/2))/(E(i)*I(i));
            end
         end
      end
   end
end

% Momento en barras con una carga puntual
for i =1:N(1,1)
   for j=1:101
      if Pp(i,1)~=0
        if (xL(i,j)/L(i))< Pp(i,2)
          Vp(i,j)= (u(i,3)*xL(i,j)) +(-((Fg1(i,2)*(xL(i,j)^3))/3)+
((Fg1(i,3)*(xL(i,j)^2))/2))/(E(i)*I(i));
         else
```

```
          Vp(i,j)=(u(i,3)*xL(i,j))  +(-(((Pp(i,1)*xL(i,j)^3)/3)-
((Pp(i,1)*(Pp(i,2)*xL(i,j))*xL(i,j)^2)/2)) -((Fg1(i,2)*(xL(i,j)^3))/3)+
((Fg1(i,3)*(xL(i,j)^2))/2))/(E(i)*I(i));
        end
      else if Pp(i,1)==0
          if (xL(i,j)/L(i))< Pp(i,2)
          Vp(i,j)= (u(i,3)*xL(i,j)) +(-((Fg(i,2)*(xL(i,j)^3))/3)+
((Fg(i,3)*(xL(i,j)^2))/2))/(E(i)*I(i));
          else
          Vp(i,j)=(u(i,3)*xL(i,j))  +(-(((Pp(i,1)*xL(i,j)^3)/3)-
((Pp(i,1)*(Pp(i,2)*xL(i,j))*xL(i,j)^2)/2)) -((Fg(i,2)*(xL(i,j)^3))/3)+
((Fg(i,3)*(xL(i,j)^2))/2))/(E(i)*I(i));
          end
          end
     end
  end
end

for i =1:N(1,1)
   for j=1:101
      if Pp(:,:)==0
          Vp(i,j)=0;
      end
   end
end


global V

V=Vd+Vt+Vp;


% Calculo de la flecha producida por Pd teniendo en cuenta los
% desplazamientos nodales
for i = 1:N(1,1)
   for j=1:101
      if Pd(:,:)==0
          Vddef(i,j)=0;
      else if Pd(i)~=0
          Vddef(i,j)= -(u(i,3)*xLdef(i,j)) +((+Pd(i)*(xLdef(i,j)^4))/8*(E(i)*I(i)))-
((Fg1(i,2)*(xLdef(i,j)^3))/3*(E(i)*I(i)))+((Fg1(i,3)*(xLdef(i,j)^2))/2*(E(i)*I(i))) ;
          else if Pd(i)==0
                Vddef(i,j)= -(u(i,3)*xLdef(i,j)) + ((Pd(i)*(xLdef(i,j)^4))/8*(E(i)*I(i)))-
((Fg1(i,2)*(xLdef(i,j)^3))/3*E(i)*I(i))+((Fg1(i,3)*(xLdef(i,j)^2))/2*(E(i)*I(i)))  ;
             end
          end
      end
   end
```

```
end
% Calculo del momento producido por una carga triangular con
% desplazamientos nodales
for i=1:N(1,1)
   for j=1:101
      if Pt(:,:)==0
        Vtdef(i,j)=0;
      else if Pt(i)~=0
          Vtdef(i,j)= (u(i,3)*xLdef(i,j)) +(-((Pt(i)*xLdef(i,j)^5)/(30*Ldef(i)))-
((Fg1(i,2)*(xLdef(i,j)^3))/3)+ ((Fg1(i,3)*(xLdef(i,j)^2))/2))/(E(i)*I(i));
         else if Pt(i)==0
              Vtdef(i,j)= (u(i,3)*xLdef(i,j)) +(-((Pt(i)*xLdef(i,j)^5)/(30*Ldef(i)))-
((Fg(i,2)*(xLdef(i,j)^3))/3)+ ((Fg(i,3)*(xLdef(i,j)^2))/2))/(E(i)*I(i));
            end
         end
      end
   end
end

% Momento en barras con una carga puntual
for i =1:N(1,1)
   for j=1:101
      if Pp(i,1)~=0
        if (xL(i,j)/Ldef(i))< Pp(i,2)
         Vpdef(i,j)= (u(i,3)*xLdef(i,j)) +(-((Fg1(i,2)*(xLdef(i,j)^3))/3)+
((Fg1(i,3)*(xLdef(i,j)^2))/2))/(E(i)*I(i));
         else
         Vpdef(i,j)=(u(i,3)*xLdef(i,j))  +(-(((Pp(i,1)*xLdef(i,j)^3)/3)-
((Pp(i,1)*(Pp(i,2)*xLdef(i,j))*xLdef(i,j)^2)/2)) -((Fg1(i,2)*(xLdef(i,j)^3))/3)+
((Fg1(i,3)*(xLdef(i,j)^2))/2))/(E(i)*I(i));
        end
      else if Pp(i,1)==0
          if (xL(i,j)/Ldef(i))< Pp(i,2)
          Vpdef(i,j)= (u(i,3)*xLdef(i,j)) +(-((Fg(i,2)*(xLdef(i,j)^3))/3)+
((Fg(i,3)*(xLdef(i,j)^2))/2))/(E(i)*I(i));
          else
          Vpdef(i,j)=(u(i,3)*xLdef(i,j))  +(-(((Pp(i,1)*xLdef(i,j)^3)/3)-
((Pp(i,1)*(Pp(i,2)*xLdef(i,j))*xLdef(i,j)^2)/2)) -((Fg(i,2)*(xLdef(i,j)^3))/3)+
((Fg(i,3)*(xLdef(i,j)^2))/2))/(E(i)*I(i));
          end
          end
     end
  end
end

for i =1:N(1,1)
   for j=1:101
      if Pp(:,:)==0
```

```
        Vpdef(i,j)=0;
      end
   end
end


global Vdef

Vdef=Vddef+Vtdef+Vpdef;


 %  Factor de escala  flecha
 Esc=(L(1)/10)/max(max(abs(Vdef)));
 %  Escalado flecha
for i=1:N(1,1)
   VE(i,:)=Vdef(i,:)*Esc;
end
%  Giro flecha escalada
for i=1:N(1,1)
   for j=1:101
   radio(i,j)=sqrt((xLdef(i,j)^2)+(VE(i,j)^2));
   theta(i,j)= atan(VE(i,j)/xLdef(i,j)) + alfadef(i);
   xv(i,j)= barradef(i,1)+ radio(i,j)*cos(theta(i,j));
   yv(i,j)= barradef(i,2) + radio(i,j)*sin(theta(i,j));
   end
end
XV=(xbdef+xv)/2;
YV=(ybdef+yv)/2;
hold on
for i=1:N(1,1)
     plot(xb(i,:),yb(i,:),'k','linewidth',8)
     plot(XV(i,:),YV(i,:),'g','linewidth',3)
  %plot(xbdef(i,:),ybdef(i,:),'g','linewidth',3)
end
axis off
end
```

Anexo 2.

En el siguiente anexo se mostrarán los archivos que forma la Guide que fueron creados por el autor del programa UCMM y que han sido editados por necesidad del UCMMv2.0.

Menu principal.

```
function varargout = Menu_Principal(varargin)
```

```
% MENU_PRINCIPAL M-file for Menu_Principal.fig
%      MENU_PRINCIPAL, by itself, creates a new MENU_PRINCIPAL or raises the
existing
%      singleton*.
%
%      H = MENU_PRINCIPAL returns the handle to a new MENU_PRINCIPAL or the
handle to
%      the existing singleton*.
%
%      MENU_PRINCIPAL('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MENU_PRINCIPAL.M with the given input
arguments.
%
%      MENU_PRINCIPAL('Property','Value',...) creates a new MENU_PRINCIPAL or raises
the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Menu_Principal_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Menu_Principal_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Menu_Principal

% Last Modified by GUIDE v2.5 27-Jun-2012 12:46:06

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Menu_Principal_OpeningFcn, ...
                   'gui_OutputFcn',  @Menu_Principal_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before Menu_Principal is made visible.
function Menu_Principal_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Menu_Principal (see VARARGIN)

% Choose default command line output for Menu_Principal
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Menu_Principal wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Menu_Principal_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in desparti_menu.
function desparti_menu_Callback(hObject, eventdata, handles)
% hObject    handle to desparti_menu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Definicion_Estructura_Articulada

uiwait

% --- Executes on button press in fuerzreti_menu.
function fuerzreti_menu_Callback(hObject, eventdata, handles)
% hObject    handle to fuerzreti_menu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
FuerzasReticulada

uiwait
```

```
% --- Executes on button press in despreti_menu.
function despreti_menu_Callback(hObject, eventdata, handles)
% hObject    handle to despreti_menu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Definicion_Estructura_Reticulada


uiwait

% --- Executes on button press in fuerzarti_menu.
function fuerzarti_menu_Callback(hObject, eventdata, handles)
% hObject    handle to fuerzarti_menu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
FuerzasArticuladas

uiwait

% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clear all
clc
close Menu_Principal


% --- Executes on button press in desplazamientoarticualdas_menuppal.
function desplazamientoarticualdas_menuppal_Callback(hObject, eventdata, handles)
% hObject    handle to desplazamientoarticualdas_menuppal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Desplazamiento_Articualdas

uiwait


% --- Executes on button press in desplazamientosreticulada_menuppal.
function desplazamientosreticulada_menuppal_Callback(hObject, eventdata, handles)
% hObject    handle to desplazamientosreticulada_menuppal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Desplazamiento_Reticuladas

uiwait
```

```
% --- Executes on button press in fuerzasestaticoA_menuppal.
function fuerzasestaticoA_menuppal_Callback(hObject, eventdata, handles)
% hObject    handle to fuerzasestaticoA_menuppal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Fuerza_Estatico_Articulada

uiwait

% --- Executes on button press in fuerzasdinamicoA_menuppal.
function fuerzasdinamicoA_menuppal_Callback(hObject, eventdata, handles)
% hObject    handle to fuerzasdinamicoA_menuppal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Fuerza_Dinamica_Articulada

uiwait

% --- Executes on button press in valorespropiosA_menuppal.
function valorespropiosA_menuppal_Callback(hObject, eventdata, handles)
% hObject    handle to valorespropiosA_menuppal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global E A densidad BARRAS Nodo RestriccionesNodo Barramuelle
ValorespropiosArticulada ans

valores_propios_estructura_articulada(E,A,densidad,BARRAS,Nodo,RestriccionesNodo,
Barramuelle);

ValorespropiosArticulada=ans;

Valores_Propios_EA

uiwait

% --- Executes on button press in desplazamientosestaticoR_menuppal.
function desplazamientosestaticoR_menuppal_Callback(hObject, eventdata, handles)
% hObject    handle to desplazamientosestaticoR_menuppal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Calculo_Desplazamientos_ER_Estatico

uiwait

% --- Executes on button press in reaccionesestaticoR_menuppal.
function reaccionesestaticoR_menuppal_Callback(hObject, eventdata, handles)
% hObject    handle to reaccionesestaticoR_menuppal (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Calculo_Fuerzas_ER_Estatico

uiwait

% --- Executes on button press in desplazamientosestaticoA_menuppal.
function desplazamientosestaticoA_menuppal_Callback(hObject, eventdata, handles)
% hObject    handle to desplazamientosestaticoA_menuppal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Calculo_Desplazamientos_EA_Estatico

uiwait

% --- Executes on button press in reaccionesestaticoA_menuppal.
function reaccionesestaticoA_menuppal_Callback(hObject, eventdata, handles)
% hObject    handle to reaccionesestaticoA_menuppal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Calculo_Fuerzas_EA_Estatico

uiwait

% --- Executes on button press in fuerzasdinamicoR_menuppal.
function fuerzasdinamicoR_menuppal_Callback(hObject, eventdata, handles)
% hObject    handle to fuerzasdinamicoR_menuppal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Fuerza_Dinamica_Reticulada

uiwait
% --- Executes on button press in valorespropiosR_menuppal.
function valorespropiosR_menuppal_Callback(hObject, eventdata, handles)
% hObject    handle to valorespropiosR_menuppal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global E A I densidad BARRAS Nodo RestriccionesNodo Barramuelle
ValorespropiosReticulada ans

valores_propios_estructura_reticulada(E,A,I,densidad,Nodo,BARRAS,RestriccionesNod
o,Barramuelle)

ValorespropiosReticulada=ans;

Valores_Propios_ER

uiwait
```

```
% --- Executes on button press in fuerzasestaticoR_menuppal.
function fuerzasestaticoR_menuppal_Callback(hObject, eventdata, handles)
% hObject    handle to fuerzasestaticoR_menuppal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Fuerza_Estatico_Reticulada

uiwait


% --- Executes on button press in comdatart.
function comdatart_Callback(hObject, eventdata, handles)
% hObject    handle to comdatart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Comprobacion_Datos_EA

uiwait


% --- Executes on button press in diagramamomentos_menuppal.
function diagramamomentos_menuppal_Callback(hObject, eventdata, handles)
% hObject    handle to diagramamomentos_menuppal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Calculo_Momentos_EA

uiwait


% --- Executes on button press in diagramaflecha_menuppal.
function diagramaflecha_menuppal_Callback(hObject, eventdata, handles)
% hObject    handle to diagramaflecha_menuppal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Dibujo_Flecha_Deformada_EA

uiwait


% --- Executes on button press in diagramamomentosER_menuppal.
function diagramamomentosER_menuppal_Callback(hObject, eventdata, handles)
% hObject    handle to diagramamomentosER_menuppal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Calculo_Momentos_ER
```

```
uiwait

% --- Executes on button press in diagramadeformadaER_menuppal.
function diagramadeformadaER_menuppal_Callback(hObject, eventdata, handles)
% hObject    handle to diagramadeformadaER_menuppal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Dibujo_Flecha_Deformada_ER

uiwait
```

Especificacion de las propiedades de la sección en articuladas.

```
function varargout = Especificacion_Secciones(varargin)
% ESPECIFICACION_SECCIONES M-file for Especificacion_Secciones.fig
%      ESPECIFICACION_SECCIONES, by itself, creates a new ESPECIFICACION_SECCIONES
or raises the existing
%      singleton*.
%
%      H = ESPECIFICACION_SECCIONES returns the handle to a new
ESPECIFICACION_SECCIONES or the handle to
%      the existing singleton*.
%
%      ESPECIFICACION_SECCIONES('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in ESPECIFICACION_SECCIONES.M with the given input
arguments.
%
%      ESPECIFICACION_SECCIONES('Property','Value',...) creates a new
ESPECIFICACION_SECCIONES or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Especificacion_Secciones_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Especificacion_Secciones_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Especificacion_Secciones

% Last Modified by GUIDE v2.5 06-Nov-2011 17:19:56

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
```

```
                'gui_OpeningFcn', @Especificacion_Secciones_OpeningFcn, ...
                'gui_OutputFcn',  @Especificacion_Secciones_OutputFcn, ...
                'gui_LayoutFcn',  [] , ...
                'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Especificacion_Secciones is made visible.
function Especificacion_Secciones_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Especificacion_Secciones (see VARARGIN)

% Choose default command line output for Especificacion_Secciones
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Especificacion_Secciones wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Especificacion_Secciones_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

global numero_secciones tablacaract

set(handles.numseccion_output,'String',numero_secciones)
```

```
set(handles.uitable1,'data',tablacaract)

% --- Executes on button press in tipos_seccion.
function tipos_seccion_Callback(hObject, eventdata, handles)
% hObject    handle to tipos_seccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Tipos_de_Seccion

uiwait

global numero_secciones
set(handles.numseccion_output,'String',numero_secciones)

% --- Executes on button press in asignacion_seccion.
function asignacion_seccion_Callback(hObject, eventdata, handles)
% hObject    handle to asignacion_seccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global numero_barras seccion A E densidad barracolespecificacionsecciones
numero_secciones contadorbarseccart Ia


if numero_secciones==1
   for i=1:numero_barras
      E(i,:)=seccion(1,1);
A(i,:)=seccion(1,2);

densidad(i,:)=seccion(1,3);
Ia(i,:)=seccion(1,4);
barracolespecificacionsecciones(i,:)=i;
   end

elseif numero_secciones>1

   for i=1:numero_barras
      contadorbarseccart=i;
Asignacion_Seccion_Articuladas

uiwait

global asignacionbarra asignacionseccion
E(asignacionbarra,:)=seccion(asignacionseccion,1);
A(asignacionbarra,:)=seccion(asignacionseccion,2);

densidad(asignacionbarra,:)=seccion(asignacionseccion,3);
Ia(asignacionbarra,:)=seccion(asignacionseccion,4);
barracolespecificacionsecciones(i,:)=i;
```

```
   end

end

% --- Executes on button press in especificacion_seccion.
function especificacion_seccion_Callback(hObject, eventdata, handles)
% hObject    handle to especificacion_seccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global numero_secciones seccion tablacaract



for i=1:numero_secciones
Caracteristicas_de_Seccion_Articuladas

uiwait

global Ee Aa Ia densidadd
seccion(i,:)=[Ee Aa densidadd Ia];
Eee(i,:)=Ee;
Aaa(i,:)=Aa;
Iaa(i,:)=Ia;
densidaddd(i,:)=densidadd;
fuercolvectorarticulada=[1:1:i];
fuercolarticulada=fuercolvectorarticulada';

end

tablacaract=[fuercolarticulada Eee Aaa densidaddd Iaa];

set(handles.uitable1,'data',tablacaract)


function numseccion_output_Callback(hObject, eventdata, handles)
% hObject    handle to numseccion_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of numseccion_output as text
%        str2double(get(hObject,'String')) returns contents of numseccion_output as a
double


% --- Executes during object creation, after setting all properties.
function numseccion_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numseccion_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function seccdensidad_output_Callback(hObject, eventdata, handles)
% hObject    handle to seccdensidad_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of seccdensidad_output as text
%        str2double(get(hObject,'String')) returns contents of seccdensidad_output as a
double


% --- Executes during object creation, after setting all properties.
function seccdensidad_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to seccdensidad_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function seccA_output_Callback(hObject, eventdata, handles)
% hObject    handle to seccA_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of seccA_output as text
%        str2double(get(hObject,'String')) returns contents of seccA_output as a double


% --- Executes during object creation, after setting all properties.
function seccA_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to seccA_output (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function seccE_output_Callback(hObject, eventdata, handles)
% hObject    handle to seccE_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of seccE_output as text
%        str2double(get(hObject,'String')) returns contents of seccE_output as a double


% --- Executes during object creation, after setting all properties.
function seccE_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to seccE_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Especificacion_Secciones



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a double


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function numseccbar_output_Callback(hObject, eventdata, handles)
% hObject    handle to numseccbar_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of numseccbar_output as text
%        str2double(get(hObject,'String')) returns contents of numseccbar_output as a
double


% --- Executes during object creation, after setting all properties.
function numseccbar_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numseccbar_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11 as a double


% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

Especificación de las propiedades de la sección en reticuladas.

```
function varargout = Especificacion_Secciones_ER(varargin)
% ESPECIFICACION_SECCIONES_ER M-file for Especificacion_Secciones_ER.fig
%      ESPECIFICACION_SECCIONES_ER, by itself, creates a new
ESPECIFICACION_SECCIONES_ER or raises the existing
%      singleton*.
%
%      H = ESPECIFICACION_SECCIONES_ER returns the handle to a new
ESPECIFICACION_SECCIONES_ER or the handle to
%      the existing singleton*.
%
%      ESPECIFICACION_SECCIONES_ER('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in ESPECIFICACION_SECCIONES_ER.M with the given
input arguments.
%
%      ESPECIFICACION_SECCIONES_ER('Property','Value',...) creates a new
ESPECIFICACION_SECCIONES_ER or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Especificacion_Secciones_ER_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Especificacion_Secciones_ER_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
```

```
% Edit the above text to modify the response to help Especificacion_Secciones_ER

% Last Modified by GUIDE v2.5 18-Nov-2011 19:39:45

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Especificacion_Secciones_ER_OpeningFcn, ...
                   'gui_OutputFcn',  @Especificacion_Secciones_ER_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Especificacion_Secciones_ER is made visible.
function Especificacion_Secciones_ER_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Especificacion_Secciones_ER (see VARARGIN)

% Choose default command line output for Especificacion_Secciones_ER
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Especificacion_Secciones_ER wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Especificacion_Secciones_ER_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


global numero_materiales tablamateriales
set(handles.nummaterial_output,'String',numero_materiales)
set(handles.uitable1,'data',tablamateriales)

% --- Executes on button press in tipos_material.
function tipos_material_Callback(hObject, eventdata, handles)
% hObject    handle to tipos_material (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Tipos_Seccion_ER

uiwait

global numero_materiales
set(handles.nummaterial_output,'String',numero_materiales)

% --- Executes on button press in asignacion_material.
function asignacion_material_Callback(hObject, eventdata, handles)
% hObject    handle to asignacion_material (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global numero_barras seccion A E I densidad barracolespecificacionmateriales
numero_materiales contadorasigseccret

if numero_materiales==1
   for i=1:numero_barras
      E(i,:)=seccion(1,1);
A(i,:)=seccion(1,2);
I(i,:)=seccion(1,3);
densidad(i,:)=seccion(1,4);
barracolespecificacionmateriales(i,:)=i;
   end

elseif numero_materiales>1


   for i=1:numero_barras
      contadorasigseccret=i;
Asignacion_Seccion

uiwait
```

```
global asignacionbarra asignacionseccion
E(asignacionbarra,:)=seccion(asignacionseccion,1);
A(asignacionbarra,:)=seccion(asignacionseccion,2);
I(asignacionbarra,:)=seccion(asignacionseccion,3);
densidad(asignacionbarra,:)=seccion(asignacionseccion,4);
barracolespecificacionmateriales(i,:)=i;
   end

end


% --- Executes on button press in especificacion_material.
function especificacion_material_Callback(hObject, eventdata, handles)
% hObject    handle to especificacion_material (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global numero_materiales seccion contadorespecifseccret tablamateriales



for i=1:numero_materiales
   contadorespecifseccret=i;
Caracteristicas_de_Seccion

uiwait

global Ee Aa Ii densidadd
seccion(i,:)=[Ee Aa Ii densidadd];
Eee(i,:)=Ee;
Aaa(i,:)=Aa;
Iii(i,:)=Ii;
densidaddd(i,:)=densidadd;
fuercolvectorreticulada=[1:1:i];
fuercolreticulada=fuercolvectorreticulada';



end
tablamateriales=[fuercolreticulada Eee Aaa Iii densidaddd];



set(handles.uitable1,'data',tablamateriales)
```

```
function nummaterial_output_Callback(hObject, eventdata, handles)
% hObject    handle to nummaterial_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nummaterial_output as text
%        str2double(get(hObject,'String')) returns contents of nummaterial_output as a
double


% --- Executes during object creation, after setting all properties.
function nummaterial_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nummaterial_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function matdensidad_output_Callback(hObject, eventdata, handles)
% hObject    handle to matdensidad_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of matdensidad_output as text
%        str2double(get(hObject,'String')) returns contents of matdensidad_output as a
double


% --- Executes during object creation, after setting all properties.
function matdensidad_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to matdensidad_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

```
function matI_output_Callback(hObject, eventdata, handles)
% hObject    handle to matI_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of matI_output as text
%        str2double(get(hObject,'String')) returns contents of matI_output as a double


% --- Executes during object creation, after setting all properties.
function matI_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to matI_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function matA_output_Callback(hObject, eventdata, handles)
% hObject    handle to matA_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of matA_output as text
%        str2double(get(hObject,'String')) returns contents of matA_output as a double


% --- Executes during object creation, after setting all properties.
function matA_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to matA_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

```
function matE_output_Callback(hObject, eventdata, handles)
% hObject    handle to matE_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of matE_output as text
%        str2double(get(hObject,'String')) returns contents of matE_output as a double


% --- Executes during object creation, after setting all properties.
function matE_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to matE_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Especificacion_Secciones_ER


function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a double


% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function nummatbar_output_Callback(hObject, eventdata, handles)
% hObject    handle to nummatbar_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nummatbar_output as text
%        str2double(get(hObject,'String')) returns contents of nummatbar_output as a
double


% --- Executes during object creation, after setting all properties.
function nummatbar_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nummatbar_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12 as a double


% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

Definición de fuerzas estáticas en articuladas.

```
function varargout = Fuerza_Estatico_Articulada(varargin)
% FUERZA_ESTATICO_ARTICULADA M-file for Fuerza_Estatico_Articulada.fig
%      FUERZA_ESTATICO_ARTICULADA, by itself, creates a new
FUERZA_ESTATICO_ARTICULADA or raises the existing
%      singleton*.
%
%      H = FUERZA_ESTATICO_ARTICULADA returns the handle to a new
FUERZA_ESTATICO_ARTICULADA or the handle to
%      the existing singleton*.
%
%      FUERZA_ESTATICO_ARTICULADA('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in FUERZA_ESTATICO_ARTICULADA.M with the given
input arguments.
```

```
%
%     FUERZA_ESTATICO_ARTICULADA('Property','Value',...) creates a new
FUERZA_ESTATICO_ARTICULADA or raises the
%     existing singleton*.  Starting from the left, property value pairs are
%     applied to the GUI before Fuerza_Estatico_Articulada_OpeningFcn gets called.  An
%     unrecognized property name or invalid value makes property application
%     stop.  All inputs are passed to Fuerza_Estatico_Articulada_OpeningFcn via
varargin.
%
%     *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Fuerza_Estatico_Articulada

% Last Modified by GUIDE v2.5 16-May-2012 19:03:24

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Fuerza_Estatico_Articulada_OpeningFcn, ...
                   'gui_OutputFcn',  @Fuerza_Estatico_Articulada_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Fuerza_Estatico_Articulada is made visible.
function Fuerza_Estatico_Articulada_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Fuerza_Estatico_Articulada (see VARARGIN)

% Choose default command line output for Fuerza_Estatico_Articulada
```

```
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Fuerza_Estatico_Articulada wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Fuerza_Estatico_Articulada_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

global numero_aplicfuerzas DatosFuerzas numero_aplicfdistribuidas
DatosFuerzasDistribuida nftriangularesaplicadas DatosFuerzasTriangular
nfpuntualesaplicadas DatosFuerzasPuntuales
set(handles.numfuerzas_output,'String',numero_aplicfuerzas)
set(handles.uitable2,'data',DatosFuerzas)
set(handles.numerodistr_output,'String',numero_aplicfdistribuidas)
set(handles.uitable3,'data',DatosFuerzasDistribuida)
set(handles.numerotriangulares_output,'String',nftriangularesaplicadas)
set(handles.uitable4,'data',DatosFuerzasTriangular)
set(handles.numeropuntuales_output,'String',nfpuntualesaplicadas)
set(handles.uitable5,'data',DatosFuerzasPuntuales)

% --- Executes on button press in numeronodosfuerza_Fuerza_Estatico_Articulada.
function numeronodosfuerza_Fuerza_Estatico_Articulada_Callback(hObject,
eventdata, handles)
% hObject    handle to numeronodosfuerza_Fuerza_Estatico_Articulada (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Numero_Fuerzas_Estatico_EA

uiwait


global numero_aplicfuerzas
set(handles.numfuerzas_output,'String',numero_aplicfuerzas)


function numfuerzas_output_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to numfuerzas_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of numfuerzas_output as text
%        str2double(get(hObject,'String')) returns contents of numfuerzas_output as a
double


% --- Executes during object creation, after setting all properties.
function numfuerzas_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numfuerzas_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in sentidofuerzas_Fuerza_Estatico_Articulada.
function sentidofuerzas_Fuerza_Estatico_Articulada_Callback(hObject, eventdata,
handles)
% hObject    handle to sentidofuerzas_Fuerza_Estatico_Articulada (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global numero_nodos FuerzapuntualNodos numero_aplicfuerzas contadorfueestar
DatosFuerzas
FuerzapuntualNodos=zeros(numero_nodos,2);
fuercolvector=[1:1:numero_nodos];
fuercol=fuercolvector';


for i=1:numero_aplicfuerzas
   contadorfueestar=i;
Definicion_Fuerzas_Estatico_EA

uiwait

global numeroaplicfuerzas fuerzaaah fuerzaaav
FuerzapuntualNodos(numeroaplicfuerzas,:)=[fuerzaaah fuerzaaav];

end
DatosFuerzas=[fuercol FuerzapuntualNodos];
```

```
set(handles.uitable2,'data',DatosFuerzas)



% --- Executes on button press in nbarrasfuerzadistribuida.
function nbarrasfuerzadistribuida_Callback(hObject, eventdata, handles)
% hObject    handle to nbarrasfuerzadistribuida (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Numero_Fuerzas_Distribuidas_Estatico_EA

uiwait


global numero_aplicfdistribuidas
set(handles.numerodistr_output,'String',numero_aplicfdistribuidas)


function numerodistr_output_Callback(hObject, eventdata, handles)
% hObject    handle to numerodistr_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global numero_aplicfdistribuidas
set(handles.numerodistr_output,'String',numero_aplicfdistribuidas);
% Hints: get(hObject,'String') returns contents of numerodistr_output as text
%        str2double(get(hObject,'String')) returns contents of numerodistr_output as a
double


% --- Executes during object creation, after setting all properties.
function numerodistr_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numerodistr_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in propiedadesfdistribuidas.
function propiedadesfdistribuidas_Callback(hObject, eventdata, handles)
% hObject    handle to propiedadesfdistribuidas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
global numero_barras FuerzaDistribuida numero_aplicfdistribuidas contadorfueestard
DatosFuerzasDistribuida
FuerzaDistribuida=zeros(numero_barras,1);
fuercolvectord=[1:1:numero_barras];
fuercold=fuercolvectord';


for i=1:numero_aplicfdistribuidas
   contadorfueestard=i;
Definicion_Fuerzas_Distribuidas_Estatico_EA

uiwait

global numerobarraaplicacion CargaPd
FuerzaDistribuida(numerobarraaplicacion,:)=[CargaPd];

end

DatosFuerzasDistribuida=[fuercold FuerzaDistribuida];

set(handles.uitable3,'data',DatosFuerzasDistribuida)


% --- Executes on button press in nbarrasfuerzatriangular.
function nbarrasfuerzatriangular_Callback(hObject, eventdata, handles)

Numero_Fuerzas_Triangulares_Estatico_EA

uiwait

global nftriangularesaplicadas
set(handles.numerotriangulares_output,'String',nftriangularesaplicadas);



function numerotriangulares_output_Callback(hObject, eventdata, handles)
% hObject    handle to numerotriangulares_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% Hints: get(hObject,'String') returns contents of numerotriangulares_output as text
%        str2double(get(hObject,'String')) returns contents of numerotriangulares_output
as a double


% --- Executes during object creation, after setting all properties.
```

```
function numerotriangulares_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numerotriangulares_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in propiedadesftriangulares.
function propiedadesftriangulares_Callback(hObject, eventdata, handles)
% hObject    handle to propiedadesftriangulares (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global numero_barras FuerzaTriangular nftriangularesaplicadas contadorftriangulares
DatosFuerzasTriangular
FuerzaTriangular=zeros(numero_barras,1);
fuercolvectort=[1:1:numero_barras];
fuercolt=fuercolvectort';


for i=1:nftriangularesaplicadas
   contadorftriangulares=i;
Definicion_Fuerzas_Triangulares_Estatico_EA

uiwait

global numerobarraaplicacionPt CargaPt
FuerzaTriangular(numerobarraaplicacionPt,:)=[CargaPt];

end

DatosFuerzasTriangular=[fuercolt FuerzaTriangular];

set(handles.uitable4,'data',DatosFuerzasTriangular)

% --- Executes on button press in numerofpuntuales.
function numerofpuntuales_Callback(hObject, eventdata, handles)
% hObject    handle to numerofpuntuales (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Numero_Fuerzas_Puntuales_Estatico_EA
```

```
uiwait

global nfpuntualesaplicadas
set(handles.numeropuntuales_output,'String',nfpuntualesaplicadas);



function numeropuntuales_output_Callback(hObject, eventdata, handles)
% hObject    handle to numeropuntuales_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of numeropuntuales_output as text
%        str2double(get(hObject,'String')) returns contents of numeropuntuales_output
as a double


% --- Executes during object creation, after setting all properties.
function numeropuntuales_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numeropuntuales_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in propiedadesfpuntuales.
function propiedadesfpuntuales_Callback(hObject, eventdata, handles)
global numero_barras FuerzaPuntual nfpuntualesaplicadas contadorfpuntuales
DatosFuerzasPuntuales
FuerzaPuntual=zeros(numero_barras,2);
fuercolvectorp=[1:1:numero_barras];
fuercolp=fuercolvectorp';


for i=1:nfpuntualesaplicadas
   contadorfpuntuales=i;
Definicion_Fuerzas_Puntuales_Estatico_EA

uiwait
```

```
global numerobarraaplicacionPp CargaPp distanciaaplicacionPp
FuerzaPuntual(numerobarraaplicacionPp,:)=[CargaPp distanciaaplicacionPp];

end

DatosFuerzasPuntuales=[fuercolp FuerzaPuntual];

set(handles.uitable5,'data',DatosFuerzasPuntuales)

% --- Executes on button press in aceptar.
function aceptar_Callback(hObject, eventdata, handles)

global FuerzaNodos FuerzapuntualNodos Nodo BARRAS FuerzaDistribuida
FuerzaTriangular FuerzaPuntual Fdpt R Rp Rd Rt



Fdpt=Calculo_Reacciones_Barras_EA(Nodo,BARRAS,FuerzaDistribuida,FuerzaTriangula
r,FuerzaPuntual);

FuerzaNodos=FuerzapuntualNodos + Fdpt;


close Fuerza_Estatico_Articulada
```

Definición de fuerzas estáticas en estructuras rticuladas.

```
function varargout = Fuerza_Estatico_Reticulada(varargin)
% FUERZA_ESTATICO_RETICULADA M-file for Fuerza_Estatico_Reticulada.fig
%      FUERZA_ESTATICO_RETICULADA, by itself, creates a new
FUERZA_ESTATICO_RETICULADA or raises the existing
%      singleton*.
%
%      H = FUERZA_ESTATICO_RETICULADA returns the handle to a new
FUERZA_ESTATICO_RETICULADA or the handle to
%      the existing singleton*.
%
%      FUERZA_ESTATICO_RETICULADA('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in FUERZA_ESTATICO_RETICULADA.M with the given
input arguments.
%
%      FUERZA_ESTATICO_RETICULADA('Property','Value',...) creates a new
FUERZA_ESTATICO_RETICULADA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Fuerza_Estatico_Reticulada_OpeningFcn gets called.
An
```

```
%     unrecognized property name or invalid value makes property application
%     stop.  All inputs are passed to Fuerza_Estatico_Reticulada_OpeningFcn via
varargin.
%
%     *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Fuerza_Estatico_Reticulada

% Last Modified by GUIDE v2.5 15-Jun-2012 19:43:52

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Fuerza_Estatico_Reticulada_OpeningFcn, ...
                   'gui_OutputFcn',  @Fuerza_Estatico_Reticulada_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Fuerza_Estatico_Reticulada is made visible.
function Fuerza_Estatico_Reticulada_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Fuerza_Estatico_Reticulada (see VARARGIN)

% Choose default command line output for Fuerza_Estatico_Reticulada
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes Fuerza_Estatico_Reticulada wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Fuerza_Estatico_Reticulada_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

global numero_aplicfuerzas DatosFuerzas numero_fuerzasdistribuidasR
DatosFuerzasDistribuida numero_fuerzastriangularesR DatosFuerzasTriangular
numero_fuerzaspuntualesR DatosFuerzasPuntual
set(handles.numerofuerzas_output,'String',numero_aplicfuerzas)
set(handles.fuerzas_output,'data',DatosFuerzas)
set(handles.numerofuerzasdistribuidas,'String',numero_fuerzasdistribuidasR)
set(handles.uitable2,'data',DatosFuerzasDistribuida)
set(handles.numerofuerzastriangulares,'String',numero_fuerzastriangularesR)
set(handles.uitable3,'data',DatosFuerzasTriangular)
set(handles.numerofuerzaspuntuales,'String',numero_fuerzaspuntualesR)
set(handles.uitable4,'data',DatosFuerzasPuntual)

% --- Executes on button press in numeronodosFuerza_Estatico_Reticulada.
function numeronodosFuerza_Estatico_Reticulada_Callback(hObject, eventdata,
handles)
% hObject    handle to numeronodosFuerza_Estatico_Reticulada (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Numero_Fuerzas_Estatico_ER

uiwait


global numero_aplicfuerzas
set(handles.numerofuerzas_output,'String',numero_aplicfuerzas)


function numerofuerzas_output_Callback(hObject, eventdata, handles)
% hObject    handle to numerofuerzas_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of numerofuerzas_output as text
```

```
%       str2double(get(hObject,'String')) returns contents of numerofuerzas_output as a
double


% --- Executes during object creation, after setting all properties.
function numerofuerzas_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numerofuerzas_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in magnitudfuerzaFuerza_Estatico_Reticulada.
function magnitudfuerzaFuerza_Estatico_Reticulada_Callback(hObject, eventdata,
handles)
% hObject    handle to magnitudfuerzaFuerza_Estatico_Reticulada (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global numero_nodos FuerzapuntualNodos numero_aplicfuerzas contadorfueestre
DatosFuerzas
FuerzapuntualNodos=zeros(numero_nodos,3);
fuercolvector=[1:1:numero_nodos];
fuercol=fuercolvector';


for i=1:numero_aplicfuerzas
   contadorfueestre=i;
Definicion_Fuerzas_Estatico_ER

uiwait

global numeroaplicfuerzas fuerzaaah fuerzaaav momento
FuerzapuntualNodos(numeroaplicfuerzas,:)=[fuerzaaah fuerzaaav momento];

end

DatosFuerzas=[fuercol FuerzapuntualNodos];

set(handles.fuerzas_output,'data',DatosFuerzas)


% --- Executes on button press in ventananumerodistribuidas.
```

```
function ventananumerodistribuidas_Callback(hObject, eventdata, handles)
% hObject    handle to ventananumerodistribuidas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Numero_Fuerzas_Distribuidas_Estatico_ER

uiwait


global numero_fuerzasdistribuidasR
set(handles.numerofuerzasdistribuidas,'String',numero_fuerzasdistribuidasR)


function numerofuerzasdistribuidas_Callback(hObject, eventdata, handles)
% hObject    handle to numerofuerzasdistribuidas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of numerofuerzasdistribuidas as text
%        str2double(get(hObject,'String')) returns contents of numerofuerzasdistribuidas
as a double


% --- Executes during object creation, after setting all properties.
function numerofuerzasdistribuidas_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numerofuerzasdistribuidas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in propiedadesdistribuidas.
function propiedadesdistribuidas_Callback(hObject, eventdata, handles)
% hObject    handle to propiedadesdistribuidas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global numero_barras FuerzaDistribuida numero_fuerzasdistribuidasR
contadorfueestardR DatosFuerzasDistribuida
FuerzaDistribuida=zeros(numero_barras,1);
fuercolvector=[1:1:numero_barras];
fuercol=fuercolvector';
```

```
for i=1:numero_fuerzasdistribuidasR
   contadorfueestardR=i;
Definicion_Fuerzas_Distribuidas_Estatico_ER

uiwait

global  CargaPdR numerobarraaplicacion
FuerzaDistribuida(numerobarraaplicacion,:)=[CargaPdR];

end

DatosFuerzasDistribuida=[fuercol FuerzaDistribuida];

set(handles.uitable2,'data',DatosFuerzasDistribuida)


% --- Executes on button press in numerotriangulares.
function numerotriangulares_Callback(hObject, eventdata, handles)
% hObject    handle to numerotriangulares (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Numero_Fuerzas_Triangulares_Estatico_ER

uiwait


global numero_fuerzastriangularesR
set(handles.numerofuerzastriangulares,'String',numero_fuerzastriangularesR)


function numerofuerzastriangulares_Callback(hObject, eventdata, handles)
% hObject    handle to numerofuerzastriangulares (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of numerofuerzastriangulares as text
%        str2double(get(hObject,'String')) returns contents of numerofuerzastriangulares
as a double


% --- Executes during object creation, after setting all properties.
function numerofuerzastriangulares_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numerofuerzastriangulares (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in propiedadestriangulares.
function propiedadestriangulares_Callback(hObject, eventdata, handles)
% hObject    handle to propiedadestriangulares (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global numero_barras FuerzaTriangular numero_fuerzastriangularesR
contadorfuerzatriangularR DatosFuerzasTriangular
FuerzaTriangular=zeros(numero_barras,1);
fuercolvector=[1:1:numero_barras];
fuercol=fuercolvector';


for i=1:numero_fuerzastriangularesR
   contadorfuerzatriangularR=i;
Definicion_Fuerzas_Triangulares_Estatico_ER

uiwait

global  CargaPtR numerobarraaplicacionT
FuerzaTriangular(numerobarraaplicacionT,:)=[CargaPtR];

end

DatosFuerzasTriangular=[fuercol FuerzaTriangular];

set(handles.uitable3,'data',DatosFuerzasTriangular)

% --- Executes on button press in numeropuntuales.
function numeropuntuales_Callback(hObject, eventdata, handles)
% hObject    handle to numeropuntuales (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Numero_Fuerzas_Puntuales_Estatico_ER

uiwait


global numero_fuerzaspuntualesR
set(handles.numerofuerzaspuntuales,'String',numero_fuerzaspuntualesR)
```

```
function numerofuerzaspuntuales_Callback(hObject, eventdata, handles)
% hObject    handle to numerofuerzaspuntuales (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of numerofuerzaspuntuales as text
%        str2double(get(hObject,'String')) returns contents of numerofuerzaspuntuales as
a double


% --- Executes during object creation, after setting all properties.
function numerofuerzaspuntuales_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numerofuerzaspuntuales (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in propiedadespuntuales.
function propiedadespuntuales_Callback(hObject, eventdata, handles)
% hObject    handle to propiedadespuntuales (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global numero_barras FuerzaPuntual numero_fuerzaspuntualesR
contadorfuerzapuntualR DatosFuerzasPuntual
FuerzaPuntual=zeros(numero_barras,2);
fuercolvector=[1:1:numero_barras];
fuercol=fuercolvector';


for i=1:numero_fuerzaspuntualesR
   contadorfuerzapuntualR=i;
Definicion_Fuerzas_Puntuales_Estatico_ER

uiwait

global  CargaPpR numerobarraaplicacionP Distanciaa
FuerzaPuntual(numerobarraaplicacionP,:)=[CargaPpR Distanciaa];

end
```

```
DatosFuerzasPuntual=[fuercol FuerzaPuntual];

set(handles.uitable4,'data',DatosFuerzasPuntual)
% --- Executes on button press in aceptar.
function aceptar_Callback(hObject, eventdata, handles)
% hObject    handle to aceptar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global FuerzaNodos FuerzapuntualNodos Nodo BARRAS FuerzaDistribuida
FuerzaTriangular FuerzaPuntual FdptR R Rp Rd Rt

FdptR=Calculo_Reacciones_ER(Nodo,BARRAS,FuerzaDistribuida,FuerzaTriangular,Fuerz
aPuntual);

FuerzaNodos=FuerzapuntualNodos + FdptR;


close Fuerza_Estatico_Reticulada
```

Anexo 3.

En el siguiente apartado, se inlcluirá el código de las partes de la interfaz qe tuvieron que ser añadidas, siendo creadas desde 0.

Número de fuerzas distribuidas en estático.

```
function varargout = Numero_Fuerzas_Distribuidas_Estatico_EA(varargin)
% NUMERO_FUERZAS_DISTRIBUIDAS_ESTATICO_EA M-file for
Numero_Fuerzas_Distribuidas_Estatico_EA.fig
%      NUMERO_FUERZAS_DISTRIBUIDAS_ESTATICO_EA, by itself, creates a new
NUMERO_FUERZAS_DISTRIBUIDAS_ESTATICO_EA or raises the existing
%      singleton*.
%
%      H = NUMERO_FUERZAS_DISTRIBUIDAS_ESTATICO_EA returns the handle to a new
NUMERO_FUERZAS_DISTRIBUIDAS_ESTATICO_EA or the handle to
%      the existing singleton*.
%
%
NUMERO_FUERZAS_DISTRIBUIDAS_ESTATICO_EA('CALLBACK',hObject,eventData,hand
les,...) calls the local
%      function named CALLBACK in NUMERO_FUERZAS_DISTRIBUIDAS_ESTATICO_EA.M
with the given input arguments.
%
%      NUMERO_FUERZAS_DISTRIBUIDAS_ESTATICO_EA('Property','Value',...) creates a
new NUMERO_FUERZAS_DISTRIBUIDAS_ESTATICO_EA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before
Numero_Fuerzas_Distribuidas_Estatico_EA_OpeningFcn gets called.  An
```

```
%     unrecognized property name or invalid value makes property application
%     stop.  All inputs are passed to
Numero_Fuerzas_Distribuidas_Estatico_EA_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
Numero_Fuerzas_Distribuidas_Estatico_EA

% Last Modified by GUIDE v2.5 08-May-2012 19:50:45

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@Numero_Fuerzas_Distribuidas_Estatico_EA_OpeningFcn, ...
                   'gui_OutputFcn',  @Numero_Fuerzas_Distribuidas_Estatico_EA_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Numero_Fuerzas_Distribuidas_Estatico_EA is made visible.
function Numero_Fuerzas_Distribuidas_Estatico_EA_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Numero_Fuerzas_Distribuidas_Estatico_EA
(see VARARGIN)

% Choose default command line output for Numero_Fuerzas_Distribuidas_Estatico_EA
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Numero_Fuerzas_Distribuidas_Estatico_EA wait for user response
(see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Numero_Fuerzas_Distribuidas_Estatico_EA_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function nbarras_input_Callback(hObject, eventdata, handles)
% hObject    handle to nbarras_input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nbarras_input as text
%        str2double(get(hObject,'String')) returns contents of nbarras_input as a double


% --- Executes during object creation, after setting all properties.
function nbarras_input_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nbarras_input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in aceptar.
function aceptar_Callback(hObject, eventdata, handles)
global numero_aplicfdistribuidas
```

```
numero_aplicfdistribuidas=str2double(get(handles.nbarras_input,'String'));
set(handles.nbarras_output,'String',numero_aplicfdistribuidas);
% hObject    handle to aceptar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



function nbarras_output_Callback(hObject, eventdata, handles)
% hObject    handle to nbarras_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nbarras_output as text
%        str2double(get(hObject,'String')) returns contents of nbarras_output as a double


% --- Executes during object creation, after setting all properties.
function nbarras_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nbarras_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in Cerrar.
function Cerrar_Callback(hObject, eventdata, handles)
% hObject    handle to Cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Numero_Fuerzas_Distribuidas_Estatico_EA
```

Número de fuerzas distribuidas en reticuladas.

```
function varargout = Numero_Fuerzas_Distribuidas_Estatico_ER(varargin)
% NUMERO_FUERZAS_DISTRIBUIDAS_ESTATICO_ER M-file for
Numero_Fuerzas_Distribuidas_Estatico_ER.fig
%      NUMERO_FUERZAS_DISTRIBUIDAS_ESTATICO_ER, by itself, creates a new
NUMERO_FUERZAS_DISTRIBUIDAS_ESTATICO_ER or raises the existing
%      singleton*.
%
%      H = NUMERO_FUERZAS_DISTRIBUIDAS_ESTATICO_ER returns the handle to a new
NUMERO_FUERZAS_DISTRIBUIDAS_ESTATICO_ER or the handle to
```

```
%     the existing singleton*.
%
%
NUMERO_FUERZAS_DISTRIBUIDAS_ESTATICO_ER('CALLBACK',hObject,eventData,hand
les,...) calls the local
%     function named CALLBACK in NUMERO_FUERZAS_DISTRIBUIDAS_ESTATICO_ER.M
with the given input arguments.
%
%     NUMERO_FUERZAS_DISTRIBUIDAS_ESTATICO_ER('Property','Value',...) creates a
new NUMERO_FUERZAS_DISTRIBUIDAS_ESTATICO_ER or raises the
%     existing singleton*.  Starting from the left, property value pairs are
%     applied to the GUI before
Numero_Fuerzas_Distribuidas_Estatico_ER_OpeningFcn gets called.  An
%     unrecognized property name or invalid value makes property application
%     stop.  All inputs are passed to
Numero_Fuerzas_Distribuidas_Estatico_ER_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
Numero_Fuerzas_Distribuidas_Estatico_ER

% Last Modified by GUIDE v2.5 12-Jun-2012 10:55:23

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@Numero_Fuerzas_Distribuidas_Estatico_ER_OpeningFcn, ...
                   'gui_OutputFcn',  @Numero_Fuerzas_Distribuidas_Estatico_ER_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before Numero_Fuerzas_Distribuidas_Estatico_ER is made visible.
function Numero_Fuerzas_Distribuidas_Estatico_ER_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Numero_Fuerzas_Distribuidas_Estatico_ER
(see VARARGIN)

% Choose default command line output for Numero_Fuerzas_Distribuidas_Estatico_ER
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Numero_Fuerzas_Distribuidas_Estatico_ER wait for user response
(see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Numero_Fuerzas_Distribuidas_Estatico_ER_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function numerodistribuidas_input_Callback(hObject, eventdata, handles)
% hObject    handle to numerodistribuidas_input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of numerodistribuidas_input as text
%        str2double(get(hObject,'String')) returns contents of numerodistribuidas_input
as a double


% --- Executes during object creation, after setting all properties.
function numerodistribuidas_input_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numerodistribuidas_input (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in aceptar.
function aceptar_Callback(hObject, eventdata, handles)
% hObject    handle to aceptar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global numero_fuerzasdistribuidasR
numero_fuerzasdistribuidasR=str2double(get(handles.numerodistribuidas_input,'Strin
g'));
set(handles.numerodistribuidas_output,'String',numero_fuerzasdistribuidasR);


function numerodistribuidas_output_Callback(hObject, eventdata, handles)
% hObject    handle to numerodistribuidas_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of numerodistribuidas_output as text
%        str2double(get(hObject,'String')) returns contents of numerodistribuidas_output
as a double


% --- Executes during object creation, after setting all properties.
function numerodistribuidas_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numerodistribuidas_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Numero_Fuerzas_Distribuidas_Estatico_ER
```

Propiedades de las cargas distribuidas en articuladas.

```
function varargout = Definicion_Fuerzas_Distribuidas_Estatico_EA(varargin)
% DEFINICION_FUERZAS_DISTRIBUIDAS_ESTATICO_EA M-file for
Definicion_Fuerzas_Distribuidas_Estatico_EA.fig
%      DEFINICION_FUERZAS_DISTRIBUIDAS_ESTATICO_EA, by itself, creates a new
DEFINICION_FUERZAS_DISTRIBUIDAS_ESTATICO_EA or raises the existing
%      singleton*.
%
%      H = DEFINICION_FUERZAS_DISTRIBUIDAS_ESTATICO_EA returns the handle to a
new DEFINICION_FUERZAS_DISTRIBUIDAS_ESTATICO_EA or the handle to
%      the existing singleton*.
%
%
DEFINICION_FUERZAS_DISTRIBUIDAS_ESTATICO_EA('CALLBACK',hObject,eventData,ha
ndles,...) calls the local
%      function named CALLBACK in
DEFINICION_FUERZAS_DISTRIBUIDAS_ESTATICO_EA.M with the given input
arguments.
%
%      DEFINICION_FUERZAS_DISTRIBUIDAS_ESTATICO_EA('Property','Value',...) creates
a new DEFINICION_FUERZAS_DISTRIBUIDAS_ESTATICO_EA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before
Definicion_Fuerzas_Distribuidas_Estatico_EA_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to
Definicion_Fuerzas_Distribuidas_Estatico_EA_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
Definicion_Fuerzas_Distribuidas_Estatico_EA

% Last Modified by GUIDE v2.5 08-May-2012 20:43:53

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
```

```
                'gui_OpeningFcn',
@Definicion_Fuerzas_Distribuidas_Estatico_EA_OpeningFcn, ...
                'gui_OutputFcn',
@Definicion_Fuerzas_Distribuidas_Estatico_EA_OutputFcn, ...
                'gui_LayoutFcn',  [] , ...
                'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Definicion_Fuerzas_Distribuidas_Estatico_EA is made visible.
function Definicion_Fuerzas_Distribuidas_Estatico_EA_OpeningFcn(hObject,
eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Definicion_Fuerzas_Distribuidas_Estatico_EA
(see VARARGIN)

% Choose default command line output for
Definicion_Fuerzas_Distribuidas_Estatico_EA
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Definicion_Fuerzas_Distribuidas_Estatico_EA wait for user response
(see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Definicion_Fuerzas_Distribuidas_Estatico_EA_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Get default command line output from handles structure
varargout{1} = handles.output;

global contadorfueestard
set(handles.nfuerzas,'String',contadorfueestard)


function nfuerzas_Callback(hObject, eventdata, handles)
% hObject    handle to nfuerzas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nfuerzas as text
%        str2double(get(hObject,'String')) returns contents of nfuerzas as a double


% --- Executes during object creation, after setting all properties.
function nfuerzas_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nfuerzas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function barraaplicacion_Callback(hObject, eventdata, handles)
% hObject    handle to barraaplicacion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of barraaplicacion as text
%        str2double(get(hObject,'String')) returns contents of barraaplicacion as a double


% --- Executes during object creation, after setting all properties.
function barraaplicacion_CreateFcn(hObject, eventdata, handles)
% hObject    handle to barraaplicacion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function valorcarga_Callback(hObject, eventdata, handles)
% hObject    handle to valorcarga (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of valorcarga as text
%        str2double(get(hObject,'String')) returns contents of valorcarga as a double


% --- Executes during object creation, after setting all properties.
function valorcarga_CreateFcn(hObject, eventdata, handles)
% hObject    handle to valorcarga (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in aceptar.
function aceptar_Callback(hObject, eventdata, handles)
% hObject    handle to aceptar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global CargaPd  numerobarraaplicacion

numerobarraaplicacion=str2double(get(handles.barraaplicacion,'String'));
CargaPd=str2double(get(handles.valorcarga,'String'));




close Definicion_Fuerzas_Distribuidas_Estatico_EA



% --- Executes on button press in Cerrar.
function Cerrar_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to Cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Definicion_Fuerzas_Distribuidas_Estatico_EA
```

Propiedades cargas distribuidas en reticuladas.

```
function varargout = Definicion_Fuerzas_Distribuidas_Estatico_ER(varargin)
% DEFINICION_FUERZAS_DISTRIBUIDAS_ESTATICO_ER M-file for
Definicion_Fuerzas_Distribuidas_Estatico_ER.fig
%      DEFINICION_FUERZAS_DISTRIBUIDAS_ESTATICO_ER, by itself, creates a new
DEFINICION_FUERZAS_DISTRIBUIDAS_ESTATICO_ER or raises the existing
%      singleton*.
%
%      H = DEFINICION_FUERZAS_DISTRIBUIDAS_ESTATICO_ER returns the handle to a
new DEFINICION_FUERZAS_DISTRIBUIDAS_ESTATICO_ER or the handle to
%      the existing singleton*.
%
%
DEFINICION_FUERZAS_DISTRIBUIDAS_ESTATICO_ER('CALLBACK',hObject,eventData,ha
ndles,...) calls the local
%      function named CALLBACK in
DEFINICION_FUERZAS_DISTRIBUIDAS_ESTATICO_ER.M with the given input
arguments.
%
%      DEFINICION_FUERZAS_DISTRIBUIDAS_ESTATICO_ER('Property','Value',...) creates
a new DEFINICION_FUERZAS_DISTRIBUIDAS_ESTATICO_ER or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before
Definicion_Fuerzas_Distribuidas_Estatico_ER_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to
Definicion_Fuerzas_Distribuidas_Estatico_ER_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
Definicion_Fuerzas_Distribuidas_Estatico_ER

% Last Modified by GUIDE v2.5 12-Jun-2012 11:57:36

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
```

```
                'gui_OpeningFcn',
@Definicion_Fuerzas_Distribuidas_Estatico_ER_OpeningFcn, ...
                'gui_OutputFcn',
@Definicion_Fuerzas_Distribuidas_Estatico_ER_OutputFcn, ...
                'gui_LayoutFcn',  [] , ...
                'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Definicion_Fuerzas_Distribuidas_Estatico_ER is made visible.
function Definicion_Fuerzas_Distribuidas_Estatico_ER_OpeningFcn(hObject,
eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Definicion_Fuerzas_Distribuidas_Estatico_ER
(see VARARGIN)

% Choose default command line output for
Definicion_Fuerzas_Distribuidas_Estatico_ER
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Definicion_Fuerzas_Distribuidas_Estatico_ER wait for user response
(see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Definicion_Fuerzas_Distribuidas_Estatico_ER_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Get default command line output from handles structure
varargout{1} = handles.output;
global contadorfueestardR
set(handles.nfuerzasR,'String',contadorfueestardR)


function nfuerzasR_Callback(hObject, eventdata, handles)
% hObject    handle to nfuerzasR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nfuerzasR as text
%        str2double(get(hObject,'String')) returns contents of nfuerzasR as a double


% --- Executes during object creation, after setting all properties.
function nfuerzasR_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nfuerzasR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function barraaplicacion_Callback(hObject, eventdata, handles)
% hObject    handle to barraaplicacion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of barraaplicacion as text
%        str2double(get(hObject,'String')) returns contents of barraaplicacion as a double


% --- Executes during object creation, after setting all properties.
function barraaplicacion_CreateFcn(hObject, eventdata, handles)
% hObject    handle to barraaplicacion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function valorcarga_Callback(hObject, eventdata, handles)
% hObject    handle to valorcarga (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of valorcarga as text
%        str2double(get(hObject,'String')) returns contents of valorcarga as a double


% --- Executes during object creation, after setting all properties.
function valorcarga_CreateFcn(hObject, eventdata, handles)
% hObject    handle to valorcarga (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in aceptar.
function aceptar_Callback(hObject, eventdata, handles)
% hObject    handle to aceptar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global CargaPdR  numerobarraaplicacion

numerobarraaplicacion=str2double(get(handles.barraaplicacion,'String'));
CargaPdR=str2double(get(handles.valorcarga,'String'));




close Definicion_Fuerzas_Distribuidas_Estatico_ER

% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Definicion_Fuerzas_Distribuidas_ER
```

Número de fuerzas triangulares en articuladas.

```
function varargout = Numero_Fuerzas_Triangulares_Estatico_EA(varargin)
% NUMERO_FUERZAS_TRIANGULARES_ESTATICO_EA M-file for
Numero_Fuerzas_Triangulares_Estatico_EA.fig
%      NUMERO_FUERZAS_TRIANGULARES_ESTATICO_EA, by itself, creates a new
NUMERO_FUERZAS_TRIANGULARES_ESTATICO_EA or raises the existing
%      singleton*.
%
%      H = NUMERO_FUERZAS_TRIANGULARES_ESTATICO_EA returns the handle to a
new NUMERO_FUERZAS_TRIANGULARES_ESTATICO_EA or the handle to
%      the existing singleton*.
%
%
NUMERO_FUERZAS_TRIANGULARES_ESTATICO_EA('CALLBACK',hObject,eventData,han
dles,...) calls the local
%      function named CALLBACK in
NUMERO_FUERZAS_TRIANGULARES_ESTATICO_EA.M with the given input arguments.
%
%      NUMERO_FUERZAS_TRIANGULARES_ESTATICO_EA('Property','Value',...) creates a
new NUMERO_FUERZAS_TRIANGULARES_ESTATICO_EA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before
Numero_Fuerzas_Triangulares_Estatico_EA_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to
Numero_Fuerzas_Triangulares_Estatico_EA_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
Numero_Fuerzas_Triangulares_Estatico_EA

% Last Modified by GUIDE v2.5 10-May-2012 17:01:47

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
```

```
                'gui_OpeningFcn',
@Numero_Fuerzas_Triangulares_Estatico_EA_OpeningFcn, ...
                'gui_OutputFcn',  @Numero_Fuerzas_Triangulares_Estatico_EA_OutputFcn,
...
                'gui_LayoutFcn',  [] , ...
                'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Numero_Fuerzas_Triangulares_Estatico_EA is made visible.
function Numero_Fuerzas_Triangulares_Estatico_EA_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Numero_Fuerzas_Triangulares_Estatico_EA
(see VARARGIN)

% Choose default command line output for Numero_Fuerzas_Triangulares_Estatico_EA
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Numero_Fuerzas_Triangulares_Estatico_EA wait for user response
(see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Numero_Fuerzas_Triangulares_Estatico_EA_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
```

```
varargout{1} = handles.output;



function nbarrasftriangular_input_Callback(hObject, eventdata, handles)
% hObject    handle to nbarrasftriangular_input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nbarrasftriangular_input as text
%        str2double(get(hObject,'String')) returns contents of nbarrasftriangular_input as
a double


% --- Executes during object creation, after setting all properties.
function nbarrasftriangular_input_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nbarrasftriangular_input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in aceptar.
function aceptar_Callback(hObject, eventdata, handles)

global nftriangularesaplicadas
nftriangularesaplicadas=str2double(get(handles.nbarrasftriangular_input,'String'));
set(handles.nbarrasftriangular_output,'String',nftriangularesaplicadas);


function nbarrasftriangular_output_Callback(hObject, eventdata, handles)
% hObject    handle to nbarrasftriangular_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nbarrasftriangular_output as text
%        str2double(get(hObject,'String')) returns contents of nbarrasftriangular_output
as a double


% --- Executes during object creation, after setting all properties.
function nbarrasftriangular_output_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to nbarrasftriangular_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLticoAB
% handles    structure with handles and user data (see GUIDATA)
close Numero_Fuerzas_Triangulares_Estatico_EA
```

Número de fuerzas triangulares en reticuladas.

```
function varargout = Numero_Fuerzas_Triangulares_Estatico_ER(varargin)
% NUMERO_FUERZAS_TRIANGULARES_ESTATICO_ER M-file for
Numero_Fuerzas_Triangulares_Estatico_ER.fig
%      NUMERO_FUERZAS_TRIANGULARES_ESTATICO_ER, by itself, creates a new
NUMERO_FUERZAS_TRIANGULARES_ESTATICO_ER or raises the existing
%      singleton*.
%
%      H = NUMERO_FUERZAS_TRIANGULARES_ESTATICO_ER returns the handle to a
new NUMERO_FUERZAS_TRIANGULARES_ESTATICO_ER or the handle to
%      the existing singleton*.
%
%
NUMERO_FUERZAS_TRIANGULARES_ESTATICO_ER('CALLBACK',hObject,eventData,han
dles,...) calls the local
%      function named CALLBACK in
NUMERO_FUERZAS_TRIANGULARES_ESTATICO_ER.M with the given input arguments.
%
%      NUMERO_FUERZAS_TRIANGULARES_ESTATICO_ER('Property','Value',...) creates a
new NUMERO_FUERZAS_TRIANGULARES_ESTATICO_ER or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before
Numero_Fuerzas_Triangulares_Estatico_ER_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to
Numero_Fuerzas_Triangulares_Estatico_ER_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
```

```
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
Numero_Fuerzas_Triangulares_Estatico_ER

% Last Modified by GUIDE v2.5 12-Jun-2012 17:34:48

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@Numero_Fuerzas_Triangulares_Estatico_ER_OpeningFcn, ...
                   'gui_OutputFcn',  @Numero_Fuerzas_Triangulares_Estatico_ER_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Numero_Fuerzas_Triangulares_Estatico_ER is made visible.
function Numero_Fuerzas_Triangulares_Estatico_ER_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Numero_Fuerzas_Triangulares_Estatico_ER
(see VARARGIN)

% Choose default command line output for Numero_Fuerzas_Triangulares_Estatico_ER
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Numero_Fuerzas_Triangulares_Estatico_ER wait for user response
(see UIRESUME)
```

```
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Numero_Fuerzas_Triangulares_Estatico_ER_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function nbarrasftriangular_input_Callback(hObject, eventdata, handles)
% hObject    handle to nbarrasftriangular_input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nbarrasftriangular_input as text
%        str2double(get(hObject,'String')) returns contents of nbarrasftriangular_input as
a double


% --- Executes during object creation, after setting all properties.
function nbarrasftriangular_input_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nbarrasftriangular_input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in acpetar.
function acpetar_Callback(hObject, eventdata, handles)
% hObject    handle to acpetar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global numero_fuerzastriangularesR
numero_fuerzastriangularesR=str2double(get(handles.nbarrasftriangular_input,'String'
));
```

```
set(handles.nbarrasftriangular_output,'String',numero_fuerzastriangularesR);


function nbarrasftriangular_output_Callback(hObject, eventdata, handles)
% hObject    handle to nbarrasftriangular_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nbarrasftriangular_output as text
%        str2double(get(hObject,'String')) returns contents of nbarrasftriangular_output
as a double


% --- Executes during object creation, after setting all properties.
function nbarrasftriangular_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nbarrasftriangular_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Numero_Fuerzas_Triangulares_Estatico_ER
Propiedades fuerzas triangulares en articuladas.

function varargout = Definicion_Fuerzas_Triangulares_Estatico_EA(varargin)
% DEFINICION_FUERZAS_TRIANGULARES_ESTATICO_EA M-file for
Definicion_Fuerzas_Triangulares_Estatico_EA.fig
%      DEFINICION_FUERZAS_TRIANGULARES_ESTATICO_EA, by itself, creates a new
DEFINICION_FUERZAS_TRIANGULARES_ESTATICO_EA or raises the existing
%      singleton*.
%
%      H = DEFINICION_FUERZAS_TRIANGULARES_ESTATICO_EA returns the handle to a
new DEFINICION_FUERZAS_TRIANGULARES_ESTATICO_EA or the handle to
%      the existing singleton*.
%
%
DEFINICION_FUERZAS_TRIANGULARES_ESTATICO_EA('CALLBACK',hObject,eventData,h
andles,...) calls the local
```

```
%     function named CALLBACK in
DEFINICION_FUERZAS_TRIANGULARES_ESTATICO_EA.M with the given input
arguments.
%
%     DEFINICION_FUERZAS_TRIANGULARES_ESTATICO_EA('Property','Value',...)
creates a new DEFINICION_FUERZAS_TRIANGULARES_ESTATICO_EA or raises the
%     existing singleton*.  Starting from the left, property value pairs are
%     applied to the GUI before
Definicion_Fuerzas_Triangulares_Estatico_EA_OpeningFcn gets called.  An
%     unrecognized property name or invalid value makes property application
%     stop.  All inputs are passed to
Definicion_Fuerzas_Triangulares_Estatico_EA_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
Definicion_Fuerzas_Triangulares_Estatico_EA

% Last Modified by GUIDE v2.5 10-May-2012 18:03:27

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@Definicion_Fuerzas_Triangulares_Estatico_EA_OpeningFcn, ...
                   'gui_OutputFcn',
@Definicion_Fuerzas_Triangulares_Estatico_EA_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Definicion_Fuerzas_Triangulares_Estatico_EA is made visible.
function Definicion_Fuerzas_Triangulares_Estatico_EA_OpeningFcn(hObject,
eventdata, handles, varargin)
```

```
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Definicion_Fuerzas_Triangulares_Estatico_EA
(see VARARGIN)

% Choose default command line output for
Definicion_Fuerzas_Triangulares_Estatico_EA
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Definicion_Fuerzas_Triangulares_Estatico_EA wait for user response
(see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout =
Definicion_Fuerzas_Triangulares_Estatico_EA_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

global contadorftriangulares
set(handles.contadorftriagular,'String',contadorftriangulares)



function contadorftriagular_Callback(hObject, eventdata, handles)
% hObject    handle to contadorftriagular (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of contadorftriagular as text
%        str2double(get(hObject,'String')) returns contents of contadorftriagular as a
double


% --- Executes during object creation, after setting all properties.
function contadorftriagular_CreateFcn(hObject, eventdata, handles)
% hObject    handle to contadorftriagular (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function nbarraaplicacion_Callback(hObject, eventdata, handles)
% hObject    handle to nbarraaplicacion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nbarraaplicacion as text
%       str2double(get(hObject,'String')) returns contents of nbarraaplicacion as a
double


% --- Executes during object creation, after setting all properties.
function nbarraaplicacion_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nbarraaplicacion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function valorcargaPt_Callback(hObject, eventdata, handles)
% hObject    handle to valorcargaPt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of valorcargaPt as text
%       str2double(get(hObject,'String')) returns contents of valorcargaPt as a double


% --- Executes during object creation, after setting all properties.
function valorcargaPt_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to valorcargaPt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in aceptar.
function aceptar_Callback(hObject, eventdata, handles)
% hObject    handle to aceptar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global CargaPt numerobarraaplicacionPt

numerobarraaplicacionPt=str2double(get(handles.nbarraaplicacion,'String'));
CargaPt=str2double(get(handles.valorcargaPt,'String'));



close Definicion_Fuerzas_Triangulares_Estatico_EA


% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Definicion_Fuerzas_Triangulares_Estatico_EA
```

Propiedades fuerzas triangulares en reticuladas.

```
function varargout = Definicion_Fuerzas_Triangulares_Estatico_ER(varargin)
% DEFINICION_FUERZAS_TRIANGULARES_ESTATICO_ER M-file for
Definicion_Fuerzas_Triangulares_Estatico_ER.fig
%      DEFINICION_FUERZAS_TRIANGULARES_ESTATICO_ER, by itself, creates a new
DEFINICION_FUERZAS_TRIANGULARES_ESTATICO_ER or raises the existing
%      singleton*.
%
%      H = DEFINICION_FUERZAS_TRIANGULARES_ESTATICO_ER returns the handle to a
new DEFINICION_FUERZAS_TRIANGULARES_ESTATICO_ER or the handle to
```

```
%     the existing singleton*.
%
%
DEFINICION_FUERZAS_TRIANGULARES_ESTATICO_ER('CALLBACK',hObject,eventData,h
andles,...) calls the local
%     function named CALLBACK in
DEFINICION_FUERZAS_TRIANGULARES_ESTATICO_ER.M with the given input
arguments.
%
%     DEFINICION_FUERZAS_TRIANGULARES_ESTATICO_ER('Property','Value',...)
creates a new DEFINICION_FUERZAS_TRIANGULARES_ESTATICO_ER or raises the
%     existing singleton*.  Starting from the left, property value pairs are
%     applied to the GUI before
Definicion_Fuerzas_Triangulares_Estatico_ER_OpeningFcn gets called.  An
%     unrecognized property name or invalid value makes property application
%     stop.  All inputs are passed to
Definicion_Fuerzas_Triangulares_Estatico_ER_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
Definicion_Fuerzas_Triangulares_Estatico_ER

% Last Modified by GUIDE v2.5 12-Jun-2012 18:17:54

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@Definicion_Fuerzas_Triangulares_Estatico_ER_OpeningFcn, ...
                   'gui_OutputFcn',
@Definicion_Fuerzas_Triangulares_Estatico_ER_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before Definicion_Fuerzas_Triangulares_Estatico_ER is made visible.
function Definicion_Fuerzas_Triangulares_Estatico_ER_OpeningFcn(hObject,
eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Definicion_Fuerzas_Triangulares_Estatico_ER
(see VARARGIN)

% Choose default command line output for
Definicion_Fuerzas_Triangulares_Estatico_ER
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Definicion_Fuerzas_Triangulares_Estatico_ER wait for user response
(see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout =
Definicion_Fuerzas_Triangulares_Estatico_ER_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
global contadorfuerzatriangularR
set(handles.ncargatriangular,'String',contadorfuerzatriangularR)


function ncargatriangular_Callback(hObject, eventdata, handles)
% hObject    handle to ncargatriangular (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ncargatriangular as text
%        str2double(get(hObject,'String')) returns contents of ncargatriangular as a
double
```

```
% --- Executes during object creation, after setting all properties.
function ncargatriangular_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ncargatriangular (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function nbarra_Callback(hObject, eventdata, handles)
% hObject    handle to nbarra (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nbarra as text
%        str2double(get(hObject,'String')) returns contents of nbarra as a double


% --- Executes during object creation, after setting all properties.
function nbarra_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nbarra (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function valoraltura_Callback(hObject, eventdata, handles)
% hObject    handle to valoraltura (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of valoraltura as text
%        str2double(get(hObject,'String')) returns contents of valoraltura as a double
```

```
% --- Executes during object creation, after setting all properties.
function valoraltura_CreateFcn(hObject, eventdata, handles)
% hObject    handle to valoraltura (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in aceptar.
function aceptar_Callback(hObject, eventdata, handles)
% hObject    handle to aceptar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global CargaPtR  numerobarraaplicacionT

numerobarraaplicacionT=str2double(get(handles.nbarra,'String'));
CargaPtR=str2double(get(handles.valoraltura,'String'));




close Definicion_Fuerzas_Triangulares_Estatico_ER

% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Definicion_Fuerzas_Triangulares_Estatico_ER
Número de fuerzas puntuales en articuladas.

unction varargout = Numero_Fuerzas_Puntuales_Estatico_EA(varargin)
% NUMERO_FUERZAS_PUNTUALES_ESTATICO_EA M-file for
Numero_Fuerzas_Puntuales_Estatico_EA.fig
%      NUMERO_FUERZAS_PUNTUALES_ESTATICO_EA, by itself, creates a new
NUMERO_FUERZAS_PUNTUALES_ESTATICO_EA or raises the existing
%      singleton*.
%
%      H = NUMERO_FUERZAS_PUNTUALES_ESTATICO_EA returns the handle to a new
NUMERO_FUERZAS_PUNTUALES_ESTATICO_EA or the handle to
%      the existing singleton*.
%
```

```
%
NUMERO_FUERZAS_PUNTUALES_ESTATICO_EA('CALLBACK',hObject,eventData,handle
s,...) calls the local
%     function named CALLBACK in NUMERO_FUERZAS_PUNTUALES_ESTATICO_EA.M
with the given input arguments.
%
%     NUMERO_FUERZAS_PUNTUALES_ESTATICO_EA('Property','Value',...) creates a
new NUMERO_FUERZAS_PUNTUALES_ESTATICO_EA or raises the
%     existing singleton*.  Starting from the left, property value pairs are
%     applied to the GUI before Numero_Fuerzas_Puntuales_Estatico_EA_OpeningFcn
gets called.  An
%     unrecognized property name or invalid value makes property application
%     stop.  All inputs are passed to
Numero_Fuerzas_Puntuales_Estatico_EA_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
Numero_Fuerzas_Puntuales_Estatico_EA

% Last Modified by GUIDE v2.5 14-May-2012 10:32:00

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Numero_Fuerzas_Puntuales_Estatico_EA_OpeningFcn,
...
                   'gui_OutputFcn',  @Numero_Fuerzas_Puntuales_Estatico_EA_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Numero_Fuerzas_Puntuales_Estatico_EA is made visible.
```

```
function Numero_Fuerzas_Puntuales_Estatico_EA_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Numero_Fuerzas_Puntuales_Estatico_EA (see
VARARGIN)

% Choose default command line output for Numero_Fuerzas_Puntuales_Estatico_EA
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Numero_Fuerzas_Puntuales_Estatico_EA wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Numero_Fuerzas_Puntuales_Estatico_EA_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function nbarraspuntuales_input_Callback(hObject, eventdata, handles)
% hObject    handle to nbarraspuntuales_input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nbarraspuntuales_input as text
%        str2double(get(hObject,'String')) returns contents of nbarraspuntuales_input as
a double


% --- Executes during object creation, after setting all properties.
function nbarraspuntuales_input_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nbarraspuntuales_input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in aceptar.
function aceptar_Callback(hObject, eventdata, handles)
% hObject    handle to aceptar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global nfpuntualesaplicadas
nfpuntualesaplicadas=str2double(get(handles.nbarraspuntuales_input,'String'));
set(handles.nbarraspuntuales_output,'String',nfpuntualesaplicadas)


function nbarraspuntuales_output_Callback(hObject, eventdata, handles)
% hObject    handle to nbarraspuntuales_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nbarraspuntuales_output as text
%        str2double(get(hObject,'String')) returns contents of nbarraspuntuales_output
as a double


% --- Executes during object creation, after setting all properties.
function nbarraspuntuales_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nbarraspuntuales_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
close Numero_Fuerzas_Puntuales_Estatico_EA
```

Número de fuerzas puntuales en reticuladas.

```
function varargout = Numero_Fuerzas_Puntuales_Estatico_ER(varargin)
% NUMERO_FUERZAS_PUNTUALES_ESTATICO_ER M-file for
Numero_Fuerzas_Puntuales_Estatico_ER.fig
%      NUMERO_FUERZAS_PUNTUALES_ESTATICO_ER, by itself, creates a new
NUMERO_FUERZAS_PUNTUALES_ESTATICO_ER or raises the existing
%      singleton*.
%
%      H = NUMERO_FUERZAS_PUNTUALES_ESTATICO_ER returns the handle to a new
NUMERO_FUERZAS_PUNTUALES_ESTATICO_ER or the handle to
%      the existing singleton*.
%
%
NUMERO_FUERZAS_PUNTUALES_ESTATICO_ER('CALLBACK',hObject,eventData,handle
s,...) calls the local
%      function named CALLBACK in NUMERO_FUERZAS_PUNTUALES_ESTATICO_ER.M
with the given input arguments.
%
%      NUMERO_FUERZAS_PUNTUALES_ESTATICO_ER('Property','Value',...) creates a
new NUMERO_FUERZAS_PUNTUALES_ESTATICO_ER or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Numero_Fuerzas_Puntuales_Estatico_ER_OpeningFcn
gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to
Numero_Fuerzas_Puntuales_Estatico_ER_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
Numero_Fuerzas_Puntuales_Estatico_ER

% Last Modified by GUIDE v2.5 12-Jun-2012 17:56:04

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Numero_Fuerzas_Puntuales_Estatico_ER_OpeningFcn,
...
                   'gui_OutputFcn',  @Numero_Fuerzas_Puntuales_Estatico_ER_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
```

```
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Numero_Fuerzas_Puntuales_Estatico_ER is made visible.
function Numero_Fuerzas_Puntuales_Estatico_ER_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Numero_Fuerzas_Puntuales_Estatico_ER (see
VARARGIN)

% Choose default command line output for Numero_Fuerzas_Puntuales_Estatico_ER
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Numero_Fuerzas_Puntuales_Estatico_ER wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Numero_Fuerzas_Puntuales_Estatico_ER_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function nbarrasfpuntual_input_Callback(hObject, eventdata, handles)
% hObject    handle to nbarrasfpuntual_input (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nbarrasfpuntual_input as text
%        str2double(get(hObject,'String')) returns contents of nbarrasfpuntual_input as a
double


% --- Executes during object creation, after setting all properties.
function nbarrasfpuntual_input_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nbarrasfpuntual_input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in aceptar.
function aceptar_Callback(hObject, eventdata, handles)
% hObject    handle to aceptar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global numero_fuerzaspuntualesR
numero_fuerzaspuntualesR=str2double(get(handles.nbarrasfpuntual_input,'String'));
set(handles.nbarrasfpuntual_output,'String',numero_fuerzaspuntualesR);


function nbarrasfpuntual_output_Callback(hObject, eventdata, handles)
% hObject    handle to nbarrasfpuntual_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nbarrasfpuntual_output as text
%        str2double(get(hObject,'String')) returns contents of nbarrasfpuntual_output as
a double


% --- Executes during object creation, after setting all properties.
function nbarrasfpuntual_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nbarrasfpuntual_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Numero_Fuerzas_Puntuales_Estatico_ER
```

Propiedades fuerzas puntuales en articuladas.

```
function varargout = Definicion_Fuerzas_Puntuales_Estatico_EA(varargin)
% DEFINICION_FUERZAS_PUNTUALES_ESTATICO_EA M-file for
Definicion_Fuerzas_Puntuales_Estatico_EA.fig
%      DEFINICION_FUERZAS_PUNTUALES_ESTATICO_EA, by itself, creates a new
DEFINICION_FUERZAS_PUNTUALES_ESTATICO_EA or raises the existing
%      singleton*.
%
%      H = DEFINICION_FUERZAS_PUNTUALES_ESTATICO_EA returns the handle to a
new DEFINICION_FUERZAS_PUNTUALES_ESTATICO_EA or the handle to
%      the existing singleton*.
%
%
DEFINICION_FUERZAS_PUNTUALES_ESTATICO_EA('CALLBACK',hObject,eventData,hand
les,...) calls the local
%      function named CALLBACK in
DEFINICION_FUERZAS_PUNTUALES_ESTATICO_EA.M with the given input arguments.
%
%      DEFINICION_FUERZAS_PUNTUALES_ESTATICO_EA('Property','Value',...) creates a
new DEFINICION_FUERZAS_PUNTUALES_ESTATICO_EA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before
Definicion_Fuerzas_Puntuales_Estatico_EA_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to
Definicion_Fuerzas_Puntuales_Estatico_EA_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
```

```
% Edit the above text to modify the response to help
Definicion_Fuerzas_Puntuales_Estatico_EA

% Last Modified by GUIDE v2.5 14-May-2012 11:00:43

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@Definicion_Fuerzas_Puntuales_Estatico_EA_OpeningFcn, ...
                   'gui_OutputFcn',  @Definicion_Fuerzas_Puntuales_Estatico_EA_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Definicion_Fuerzas_Puntuales_Estatico_EA is made visible.
function Definicion_Fuerzas_Puntuales_Estatico_EA_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Definicion_Fuerzas_Puntuales_Estatico_EA
(see VARARGIN)

% Choose default command line output for Definicion_Fuerzas_Puntuales_Estatico_EA
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Definicion_Fuerzas_Puntuales_Estatico_EA wait for user response
(see UIRESUME)
% uiwait(handles.figure1);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = Definicion_Fuerzas_Puntuales_Estatico_EA_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

global contadorfpuntuales
set(handles.contadorfpuntuales_output,'String',contadorfpuntuales)



function contadorfpuntuales_output_Callback(hObject, eventdata, handles)
% hObject    handle to contadorfpuntuales_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of contadorfpuntuales_output as text
%        str2double(get(hObject,'String')) returns contents of contadorfpuntuales_output
as a double


% --- Executes during object creation, after setting all properties.
function contadorfpuntuales_output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to contadorfpuntuales_output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function nbarraaplicacionPp_input_Callback(hObject, eventdata, handles)
% hObject    handle to nbarraaplicacionPp_input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nbarraaplicacionPp_input as text
```

```
%       str2double(get(hObject,'String')) returns contents of nbarraaplicacionPp_input
as a double


% --- Executes during object creation, after setting all properties.
function nbarraaplicacionPp_input_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nbarraaplicacionPp_input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function valorcargaPp_Callback(hObject, eventdata, handles)
% hObject    handle to valorcargaPp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of valorcargaPp as text
%       str2double(get(hObject,'String')) returns contents of valorcargaPp as a double


% --- Executes during object creation, after setting all properties.
function valorcargaPp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to valorcargaPp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function distanciaaplicacion_input_Callback(hObject, eventdata, handles)
% hObject    handle to distanciaaplicacion_input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of distanciaaplicacion_input as text
%        str2double(get(hObject,'String')) returns contents of distanciaaplicacion_input as
a double


% --- Executes during object creation, after setting all properties.
function distanciaaplicacion_input_CreateFcn(hObject, eventdata, handles)
% hObject    handle to distanciaaplicacion_input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in aceptar.
function aceptar_Callback(hObject, eventdata, handles)
% hObject    handle to aceptar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global numerobarraaplicacionPp CargaPp distanciaaplicacionPp

numerobarraaplicacionPp=str2double(get(handles.nbarraaplicacionPp_input,'String'));
CargaPp=str2double(get(handles.valorcargaPp,'String'));
distanciaaplicacionPp=str2double(get(handles.distanciaaplicacion_input,'String'));



close Definicion_Fuerzas_Puntuales_Estatico_EA



% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Definicion_Fuerzas_Puntuales_Estatico_EA
```

Propiedades fuerzas puntuales en reticuladas.

```
function varargout = Definicion_Fuerzas_Puntuales_Estatico_ER(varargin)
% DEFINICION_FUERZAS_PUNTUALES_ESTATICO_ER M-file for
Definicion_Fuerzas_Puntuales_Estatico_ER.fig
```

```
%     DEFINICION_FUERZAS_PUNTUALES_ESTATICO_ER, by itself, creates a new
DEFINICION_FUERZAS_PUNTUALES_ESTATICO_ER or raises the existing
%     singleton*.
%
%     H = DEFINICION_FUERZAS_PUNTUALES_ESTATICO_ER returns the handle to a
new DEFINICION_FUERZAS_PUNTUALES_ESTATICO_ER or the handle to
%     the existing singleton*.
%
%
DEFINICION_FUERZAS_PUNTUALES_ESTATICO_ER('CALLBACK',hObject,eventData,hand
les,...) calls the local
%     function named CALLBACK in
DEFINICION_FUERZAS_PUNTUALES_ESTATICO_ER.M with the given input arguments.
%
%     DEFINICION_FUERZAS_PUNTUALES_ESTATICO_ER('Property','Value',...) creates a
new DEFINICION_FUERZAS_PUNTUALES_ESTATICO_ER or raises the
%     existing singleton*.  Starting from the left, property value pairs are
%     applied to the GUI before
Definicion_Fuerzas_Puntuales_Estatico_ER_OpeningFcn gets called.  An
%     unrecognized property name or invalid value makes property application
%     stop.  All inputs are passed to
Definicion_Fuerzas_Puntuales_Estatico_ER_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
Definicion_Fuerzas_Puntuales_Estatico_ER

% Last Modified by GUIDE v2.5 15-Jun-2012 19:25:32

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@Definicion_Fuerzas_Puntuales_Estatico_ER_OpeningFcn, ...
                   'gui_OutputFcn',  @Definicion_Fuerzas_Puntuales_Estatico_ER_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
```

```
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Definicion_Fuerzas_Puntuales_Estatico_ER is made visible.
function Definicion_Fuerzas_Puntuales_Estatico_ER_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Definicion_Fuerzas_Puntuales_Estatico_ER
(see VARARGIN)

% Choose default command line output for Definicion_Fuerzas_Puntuales_Estatico_ER
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Definicion_Fuerzas_Puntuales_Estatico_ER wait for user response
(see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Definicion_Fuerzas_Puntuales_Estatico_ER_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
global contadorfuerzapuntualR
set(handles.nfuerzaPR,'String',contadorfuerzapuntualR)


function nfuerzaPR_Callback(hObject, eventdata, handles)
% hObject    handle to nfuerzaPR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nfuerzaPR as text
```

```
%       str2double(get(hObject,'String')) returns contents of nfuerzaPR as a double


% --- Executes during object creation, after setting all properties.
function nfuerzaPR_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nfuerzaPR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function numerobarraPR_Callback(hObject, eventdata, handles)
% hObject    handle to numerobarraPR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of numerobarraPR as text
%       str2double(get(hObject,'String')) returns contents of numerobarraPR as a double


% --- Executes during object creation, after setting all properties.
function numerobarraPR_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numerobarraPR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function valorcargaPR_Callback(hObject, eventdata, handles)
% hObject    handle to valorcargaPR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of valorcargaPR as text
```

```
%       str2double(get(hObject,'String')) returns contents of valorcargaPR as a double


% --- Executes during object creation, after setting all properties.
function valorcargaPR_CreateFcn(hObject, eventdata, handles)
% hObject    handle to valorcargaPR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function DistaplicacionR_Callback(hObject, eventdata, handles)
% hObject    handle to DistaplicacionR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of DistaplicacionR as text
%       str2double(get(hObject,'String')) returns contents of DistaplicacionR as a double


% --- Executes during object creation, after setting all properties.
function DistaplicacionR_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DistaplicacionR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in aceptar.
function aceptar_Callback(hObject, eventdata, handles)
% hObject    handle to aceptar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global CargaPpR  numerobarraaplicacionP Distanciaa
```

```
numerobarraaplicacionP=str2double(get(handles.numerobarraPR,'String'));
CargaPpR=str2double(get(handles.valorcargaPR,'String'));
Distanciaa=str2double(get(handles.DistaplicacionR,'String'));



close Definicion_Fuerzas_Puntuales_Estatico_ER

% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Definicion_Fuerzas_Puntuales_Estatico_ER
```

Ventana previa a la representación de momentos.

```
function varargout = Calculo_Momentos_EA(varargin)
% CALCULO_MOMENTOS_EA M-file for Calculo_Momentos_EA.fig
%      CALCULO_MOMENTOS_EA, by itself, creates a new CALCULO_MOMENTOS_EA or
raises the existing
%      singleton*.
%
%      H = CALCULO_MOMENTOS_EA returns the handle to a new
CALCULO_MOMENTOS_EA or the handle to
%      the existing singleton*.
%
%      CALCULO_MOMENTOS_EA('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in CALCULO_MOMENTOS_EA.M with the given input
arguments.
%
%      CALCULO_MOMENTOS_EA('Property','Value',...) creates a new
CALCULO_MOMENTOS_EA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Calculo_Momentos_EA_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Calculo_Momentos_EA_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Calculo_Momentos_EA

% Last Modified by GUIDE v2.5 06-Jun-2012 20:13:42
```

```
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Calculo_Momentos_EA_OpeningFcn, ...
                   'gui_OutputFcn',  @Calculo_Momentos_EA_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Calculo_Momentos_EA is made visible.
function Calculo_Momentos_EA_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Calculo_Momentos_EA (see VARARGIN)

% Choose default command line output for Calculo_Momentos_EA
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Calculo_Momentos_EA wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Calculo_Momentos_EA_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
% --- Executes on button press in calculomomentosEA.
function calculomomentosEA_Callback(hObject, eventdata, handles)
% hObject    handle to calculomomentosEA (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


Ventana_Momentos_Flectores_EA

uiwait


% --- Executes on button press in reestablecervalores.
function reestablecervalores_Callback(hObject, eventdata, handles)
% hObject    handle to reestablecervalores (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clear all
clc

% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Calculo_Momentos_EA
```

Ventana en la que se representa la ley de momentos en articuladas.

```
function varargout = Ventana_Momentos_Flectores_EA(varargin)
% VENTANA_MOMENTOS_FLECTORES_EA M-file for
Ventana_Momentos_Flectores_EA.fig
%      VENTANA_MOMENTOS_FLECTORES_EA, by itself, creates a new
VENTANA_MOMENTOS_FLECTORES_EA or raises the existing
%      singleton*.
%
%      H = VENTANA_MOMENTOS_FLECTORES_EA returns the handle to a new
VENTANA_MOMENTOS_FLECTORES_EA or the handle to
%      the existing singleton*.
%
%
VENTANA_MOMENTOS_FLECTORES_EA('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in VENTANA_MOMENTOS_FLECTORES_EA.M with the
given input arguments.
%
```

```
%     VENTANA_MOMENTOS_FLECTORES_EA('Property','Value',...) creates a new
VENTANA_MOMENTOS_FLECTORES_EA or raises the
%     existing singleton*.  Starting from the left, property value pairs are
%     applied to the GUI before Ventana_Momentos_Flectores_EA_OpeningFcn gets
called.  An
%     unrecognized property name or invalid value makes property application
%     stop.  All inputs are passed to Ventana_Momentos_Flectores_EA_OpeningFcn via
varargin.
%
%     *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
Ventana_Momentos_Flectores_EA

% Last Modified by GUIDE v2.5 16-May-2012 18:28:36

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Ventana_Momentos_Flectores_EA_OpeningFcn, ...
                   'gui_OutputFcn',  @Ventana_Momentos_Flectores_EA_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Ventana_Momentos_Flectores_EA is made visible.
function Ventana_Momentos_Flectores_EA_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Ventana_Momentos_Flectores_EA (see
VARARGIN)
```

```
% Choose default command line output for Ventana_Momentos_Flectores_EA
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);



% UIWAIT makes Ventana_Momentos_Flectores_EA wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Ventana_Momentos_Flectores_EA_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
global Nodo BARRAS FuerzaDistribuida FuerzaTriangular FuerzaPuntual ans Diagrama
Mmax N M
Calculo_Momentos_Flectores_EA(Nodo,BARRAS,FuerzaDistribuida,FuerzaTriangular,Fu
erzaPuntual);

ans = Diagrama;



Mmax=zeros(N(1,1),2);
for i = 1:N(1,1)

   Mmax(i,1)=i;
   Mmax(i,2)=max(max(abs(M(i,:))));

end

set(handles.uitable1,'Data',Mmax)



% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
% hObject    handle to cerrar (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Ventana_Momentos_Flectores_EA
```

Ventana previa al cálculo de momentos en reticuladas.

```
function varargout = Calculo_Momentos_ER(varargin)
% CALCULO_MOMENTOS_ER M-file for Calculo_Momentos_ER.fig
%      CALCULO_MOMENTOS_ER, by itself, creates a new CALCULO_MOMENTOS_ER or
raises the existing
%      singleton*.
%
%      H = CALCULO_MOMENTOS_ER returns the handle to a new
CALCULO_MOMENTOS_ER or the handle to
%      the existing singleton*.
%
%      CALCULO_MOMENTOS_ER('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in CALCULO_MOMENTOS_ER.M with the given input
arguments.
%
%      CALCULO_MOMENTOS_ER('Property','Value',...) creates a new
CALCULO_MOMENTOS_ER or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Calculo_Momentos_ER_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Calculo_Momentos_ER_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Calculo_Momentos_ER

% Last Modified by GUIDE v2.5 18-Jun-2012 11:03:04

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Calculo_Momentos_ER_OpeningFcn, ...
                   'gui_OutputFcn',  @Calculo_Momentos_ER_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Calculo_Momentos_ER is made visible.
function Calculo_Momentos_ER_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Calculo_Momentos_ER (see VARARGIN)

% Choose default command line output for Calculo_Momentos_ER
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Calculo_Momentos_ER wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Calculo_Momentos_ER_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in momentos.
function momentos_Callback(hObject, eventdata, handles)
% hObject    handle to momentos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Ventana_Momentos_Flectores_ER

uiwait

% --- Executes on button press in restablecervalores.
```

```
function restablecervalores_Callback(hObject, eventdata, handles)
% hObject    handle to restablecervalores (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clear all
clc

% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Calculo_Momentos_ER
```

Ventana en la que aparecerá representada la ley de momentos en reticuladas.

```
function varargout = Ventana_Momentos_Flectores_ER(varargin)
% VENTANA_MOMENTOS_FLECTORES_ER M-file for
Ventana_Momentos_Flectores_ER.fig
%      VENTANA_MOMENTOS_FLECTORES_ER, by itself, creates a new
VENTANA_MOMENTOS_FLECTORES_ER or raises the existing
%      singleton*.
%
%      H = VENTANA_MOMENTOS_FLECTORES_ER returns the handle to a new
VENTANA_MOMENTOS_FLECTORES_ER or the handle to
%      the existing singleton*.
%
%
VENTANA_MOMENTOS_FLECTORES_ER('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in VENTANA_MOMENTOS_FLECTORES_ER.M with the
given input arguments.
%
%      VENTANA_MOMENTOS_FLECTORES_ER('Property','Value',...) creates a new
VENTANA_MOMENTOS_FLECTORES_ER or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Ventana_Momentos_Flectores_ER_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Ventana_Momentos_Flectores_ER_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
```

```
% Edit the above text to modify the response to help
Ventana_Momentos_Flectores_ER

% Last Modified by GUIDE v2.5 20-Jun-2012 21:40:56

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Ventana_Momentos_Flectores_ER_OpeningFcn, ...
                   'gui_OutputFcn',  @Ventana_Momentos_Flectores_ER_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Ventana_Momentos_Flectores_ER is made visible.
function Ventana_Momentos_Flectores_ER_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Ventana_Momentos_Flectores_ER (see
VARARGIN)

% Choose default command line output for Ventana_Momentos_Flectores_ER
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Ventana_Momentos_Flectores_ER wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
```

```
function varargout = Ventana_Momentos_Flectores_ER_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
global Nodo BARRAS FuerzaDistribuida FuerzaTriangular FuerzaPuntual ans Diagrama
Mmax N M DesplazamientosEstaticoER RestriccionesNodo E I A M
Calculo_Leyes_Momento_Flector_ER(Nodo,BARRAS,RestriccionesNodo,FuerzaDistribui
da,FuerzaTriangular,FuerzaPuntual,DesplazamientosEstaticoER,E,A,I);

ans = Diagrama;


Mmax=zeros(N(1,1),2);
for i = 1:N(1,1)

   Mmax(i,1)=i;
   Mmax(i,2)=max(max(abs(M(i,:))));

end

set(handles.uitable1,'Data',Mmax)



% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Ventana_Momentos_Flectores_ER
```

Ventana previa a la representación de la flecha en articuladas.

```
function varargout = Dibujo_Flecha_Deformada_EA(varargin)
% DIBUJO_FLECHA_DEFORMADA_EA M-file for Dibujo_Flecha_Deformada_EA.fig
%      DIBUJO_FLECHA_DEFORMADA_EA, by itself, creates a new
DIBUJO_FLECHA_DEFORMADA_EA or raises the existing
%      singleton*.
%
%      H = DIBUJO_FLECHA_DEFORMADA_EA returns the handle to a new
DIBUJO_FLECHA_DEFORMADA_EA or the handle to
%      the existing singleton*.
%
```

```
%     DIBUJO_FLECHA_DEFORMADA_EA('CALLBACK',hObject,eventData,handles,...)
calls the local
%     function named CALLBACK in DIBUJO_FLECHA_DEFORMADA_EA.M with the given
input arguments.
%
%     DIBUJO_FLECHA_DEFORMADA_EA('Property','Value',...) creates a new
DIBUJO_FLECHA_DEFORMADA_EA or raises the
%     existing singleton*.  Starting from the left, property value pairs are
%     applied to the GUI before Dibujo_Flecha_Deformada_EA_OpeningFcn gets called.
An
%     unrecognized property name or invalid value makes property application
%     stop.  All inputs are passed to Dibujo_Flecha_Deformada_EA_OpeningFcn via
varargin.
%
%     *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Dibujo_Flecha_Deformada_EA

% Last Modified by GUIDE v2.5 29-May-2012 10:36:47

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Dibujo_Flecha_Deformada_EA_OpeningFcn, ...
                   'gui_OutputFcn',  @Dibujo_Flecha_Deformada_EA_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Dibujo_Flecha_Deformada_EA is made visible.
function Dibujo_Flecha_Deformada_EA_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Dibujo_Flecha_Deformada_EA (see
VARARGIN)

% Choose default command line output for Dibujo_Flecha_Deformada_EA
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Dibujo_Flecha_Deformada_EA wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Dibujo_Flecha_Deformada_EA_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in Deformada.
function Deformada_Callback(hObject, eventdata, handles)
% hObject    handle to Deformada (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Ventana_Dibujo_Flecha_EA

uiwait

% --- Executes on button press in borrartodo.
function borrartodo_Callback(hObject, eventdata, handles)
% hObject    handle to borrartodo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clear all
clc


% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Dibujo_Flecha_Deformada_EA
```

Ventana en la que aparecerá la deformada en articuladas.

```
function varargout = Ventana_Dibujo_Flecha_EA(varargin)
% VENTANA_DIBUJO_FLECHA_EA M-file for Ventana_Dibujo_Flecha_EA.fig
%      VENTANA_DIBUJO_FLECHA_EA, by itself, creates a new
VENTANA_DIBUJO_FLECHA_EA or raises the existing
%      singleton*.
%
%      H = VENTANA_DIBUJO_FLECHA_EA returns the handle to a new
VENTANA_DIBUJO_FLECHA_EA or the handle to
%      the existing singleton*.
%
%      VENTANA_DIBUJO_FLECHA_EA('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in VENTANA_DIBUJO_FLECHA_EA.M with the given
input arguments.
%
%      VENTANA_DIBUJO_FLECHA_EA('Property','Value',...) creates a new
VENTANA_DIBUJO_FLECHA_EA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Ventana_Dibujo_Flecha_EA_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Ventana_Dibujo_Flecha_EA_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Ventana_Dibujo_Flecha_EA

% Last Modified by GUIDE v2.5 31-May-2012 11:04:47

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Ventana_Dibujo_Flecha_EA_OpeningFcn, ...
                   'gui_OutputFcn',  @Ventana_Dibujo_Flecha_EA_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
```

```
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Ventana_Dibujo_Flecha_EA is made visible.
function Ventana_Dibujo_Flecha_EA_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Ventana_Dibujo_Flecha_EA (see VARARGIN)

% Choose default command line output for Ventana_Dibujo_Flecha_EA
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);



% UIWAIT makes Ventana_Dibujo_Flecha_EA wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Ventana_Dibujo_Flecha_EA_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
global Nodo BARRAS FuerzaDistribuida FuerzaTriangular FuerzaPuntual ans DiagramaV
Vmax N V E Ia RestriccionesNodo DesplazamientosEstaticoEA

Calculo_Flecha_EA(Nodo,BARRAS,FuerzaDistribuida,FuerzaTriangular,FuerzaPuntual,R
estriccionesNodo,DesplazamientosEstaticoEA,E,Ia);
```

```
ans = DiagramaV;



Vmax=zeros(N(1,1),2);
for i = 1:N(1,1)

   Vmax(i,1)=i;
   Vmax(i,2)=max(max(abs(V(i,:))));

end

set(handles.uitable1,'Data',Vmax)


global  numero_nodos

contadorEAed=0; %contador estructura articulada estatico desplazamientos
for i=1:numero_nodos
   for I=1:2
      if (RestriccionesNodo(i,I)==1) && (I==1) && (contadorEAed==0)
         contadorEAed=contadorEAed+1;
         matrizindicativa=(10*i)+1;
      elseif (RestriccionesNodo(i,I)==1) && (I==2) && (contadorEAed==0)
         contadorEAed=contadorEAed+1;
         matrizindicativa=(10*i)+2;
      elseif (RestriccionesNodo(i,I)==1) && (I==1) && (contadorEAed~=0)
         contadorEAed=contadorEAed+1;
         matrizindicativa(contadorEAed,:)=(10*i)+1;
      elseif (RestriccionesNodo(i,I)==1) && (I==2) && (contadorEAed~=0)
         contadorEAed=contadorEAed+1;
         matrizindicativa(contadorEAed,:)=(10*i)+2;
      elseif (RestriccionesNodo(i,I)==0)
          contadorEAed=contadorEAed;


      end
   end
end

DesplazamientosEstaticoEAtable=[matrizindicativa DesplazamientosEstaticoEA];

set(handles.uitable2,'data',DesplazamientosEstaticoEAtable)


% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Ventana_Dibujo_Flecha_EA
```

Ventana previa a la representación de la deformada en reticuladas.

```
function varargout = Dibujo_Flecha_Deformada_ER(varargin)
% DIBUJO_FLECHA_DEFORMADA_ER M-file for Dibujo_Flecha_Deformada_ER.fig
%      DIBUJO_FLECHA_DEFORMADA_ER, by itself, creates a new
DIBUJO_FLECHA_DEFORMADA_ER or raises the existing
%      singleton*.
%
%      H = DIBUJO_FLECHA_DEFORMADA_ER returns the handle to a new
DIBUJO_FLECHA_DEFORMADA_ER or the handle to
%      the existing singleton*.
%
%      DIBUJO_FLECHA_DEFORMADA_ER('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in DIBUJO_FLECHA_DEFORMADA_ER.M with the given
input arguments.
%
%      DIBUJO_FLECHA_DEFORMADA_ER('Property','Value',...) creates a new
DIBUJO_FLECHA_DEFORMADA_ER or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Dibujo_Flecha_Deformada_ER_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Dibujo_Flecha_Deformada_ER_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Dibujo_Flecha_Deformada_ER

% Last Modified by GUIDE v2.5 18-Jun-2012 11:13:58

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Dibujo_Flecha_Deformada_ER_OpeningFcn, ...
                   'gui_OutputFcn',  @Dibujo_Flecha_Deformada_ER_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
```

```
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Dibujo_Flecha_Deformada_ER is made visible.
function Dibujo_Flecha_Deformada_ER_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Dibujo_Flecha_Deformada_ER (see
VARARGIN)

% Choose default command line output for Dibujo_Flecha_Deformada_ER
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Dibujo_Flecha_Deformada_ER wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Dibujo_Flecha_Deformada_ER_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in deformadaER.
function deformadaER_Callback(hObject, eventdata, handles)
% hObject    handle to deformadaER (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Ventana_Dibujo_Flecha_ER

% --- Executes on button press in restavlecervalores.
function restavlecervalores_Callback(hObject, eventdata, handles)
% hObject    handle to restavlecervalores (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clear all
clc

% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Dibujo_Flecha_Deformada_ER
```

Ventana en la que se representará la deformada en reticuladas.

```
function varargout = Ventana_Dibujo_Flecha_ER(varargin)
% VENTANA_DIBUJO_FLECHA_ER M-file for Ventana_Dibujo_Flecha_ER.fig
%      VENTANA_DIBUJO_FLECHA_ER, by itself, creates a new
VENTANA_DIBUJO_FLECHA_ER or raises the existing
%      singleton*.
%
%      H = VENTANA_DIBUJO_FLECHA_ER returns the handle to a new
VENTANA_DIBUJO_FLECHA_ER or the handle to
%      the existing singleton*.
%
%      VENTANA_DIBUJO_FLECHA_ER('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in VENTANA_DIBUJO_FLECHA_ER.M with the given
input arguments.
%
%      VENTANA_DIBUJO_FLECHA_ER('Property','Value',...) creates a new
VENTANA_DIBUJO_FLECHA_ER or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Ventana_Dibujo_Flecha_ER_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Ventana_Dibujo_Flecha_ER_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
```

```
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Ventana_Dibujo_Flecha_ER

% Last Modified by GUIDE v2.5 18-Jun-2012 10:56:57

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Ventana_Dibujo_Flecha_ER_OpeningFcn, ...
                   'gui_OutputFcn',  @Ventana_Dibujo_Flecha_ER_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Ventana_Dibujo_Flecha_ER is made visible.
function Ventana_Dibujo_Flecha_ER_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Ventana_Dibujo_Flecha_ER (see VARARGIN)

% Choose default command line output for Ventana_Dibujo_Flecha_ER
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Ventana_Dibujo_Flecha_ER wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Ventana_Dibujo_Flecha_ER_OutputFcn(hObject, eventdata,
handles)
```

```
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
global Nodo BARRAS FuerzaDistribuida FuerzaTriangular FuerzaPuntual ans DiagramaV
Vmax N V E I RestriccionesNodo DesplazamientosEstaticoER A

Calculo_Flecha_ER(Nodo,BARRAS,FuerzaDistribuida,FuerzaTriangular,FuerzaPuntual,R
estriccionesNodo,DesplazamientosEstaticoER,E,I,A);

ans = DiagramaV;


Vmax=zeros(N(1,1),2);
for i = 1:N(1,1)

   Vmax(i,1)=i;
   Vmax(i,2)=max(max(abs(V(i,:))));

end

set(handles.uitable1,'Data',Vmax)


global  numero_nodos

contadorEAed=0; %contador estructura articulada estatico desplazamientos
for i=1:numero_nodos
   for I=1:3
      if (RestriccionesNodo(i,I)==1) && (I==1) && (contadorEAed==0)
         contadorEAed=contadorEAed+1;
         matrizindicativaR=(10*i)+1;
      elseif (RestriccionesNodo(i,I)==1) && (I==2) && (contadorEAed==0)
         contadorEAed=contadorEAed+1;
         matrizindicativaR=(10*i)+2;
         elseif (RestriccionesNodo(i,I)==1) && (I==3) && (contadorEAed==0)
         contadorEAed=contadorEAed+1;
         matrizindicativaR=(10*i)+3;
      elseif (RestriccionesNodo(i,I)==1) && (I==1) && (contadorEAed~=0)
         contadorEAed=contadorEAed+1;
         matrizindicativaR(contadorEAed,:)=(10*i)+1;
      elseif (RestriccionesNodo(i,I)==1) && (I==2) && (contadorEAed~=0)
         contadorEAed=contadorEAed+1;
         matrizindicativaR(contadorEAed,:)=(10*i)+2;
         elseif (RestriccionesNodo(i,I)==1) && (I==3) && (contadorEAed~=0)
```

```
        contadorEAed=contadorEAed+1;
        matrizindicativaR(contadorEAed,:)=(10*i)+3;
      elseif (RestriccionesNodo(i,I)==0)
         contadorEAed=contadorEAed;


    end
  end
end

DesplazamientosEstaticoERtable=[matrizindicativaR DesplazamientosEstaticoER];

set(handles.uitable2,'data',DesplazamientosEstaticoERtable)


% --- Executes on button press in cerrar.
function cerrar_Callback(hObject, eventdata, handles)
% hObject    handle to cerrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close Ventana_Dibujo_Flecha_ER
```